Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures

Co-funded by the European Commission within the Seventh Framework Programme. Grant Agreement No. RI-213107

Deliverable JRA1.1: Evaluation of current network control and management planes for multi-domain network infrastructure

Version 1.7

Dissemination Level	Public
Contractual Date of Delivery	31 July 2008
Actual Date of Delivery	31 December 2008
Editor	Cristina Cervelló-Pastor - **UPC** Robert Machado - **UPC**
Contributors	Peter Kauffman, Monika Roesler - **DFN** Ugo Monaco - **GARR** S. Figuerola, Alejandro Berna, Josep Pons - **i2CAT** Afrodite Sevasti - **GRNET** Dimitrios Kagoleras - **ICCS** Jean-Marc Uze - **Juniper** Peter Sjödin, Markus Hidell - **KTH** Lukasz Dolata - **PSNC**
Reviewers	M. Potts - **Martel** S. Figuerola - **i2CAT** M. Campanella - **GARR**

Abstract

This deliverable includes a compilation and evaluation of available control and management architectures and protocols applicable to a multilayer infrastructure in a multi-domain Virtual Network environment.

Table of Contents

1 Introduction

1.1 Purpose and Scope

The scope of this deliverable is mainly focused on the virtualisation of the resources within a network and at processing nodes. The virtualization of the FEDERICA infrastructure allows the provisioning of its available resources to users by means of FEDERICA slices. A slice is seen by the user as a real physical network under his/her domain, however it maps to a logical partition (a virtual instance) of the physical FEDERICA resources. A slice is built to exhibit to the highest degree all the principles applicable to a physical network (isolation, reproducibility, manageability, …). Currently, there are no standard definitions available for network virtualization or its associated architectures. Therefore, this deliverable proposes the Virtual Network layer architecture and evaluates a set of Management- and Control Planes that can be used for the partitioning and virtualization of the FEDERICA network resources. This evaluation has been performed taking into account an initial set of FEDERICA requirements; a possible extension of the selected tools will be evaluated in future deliverables.

The studies described in this deliverable define the virtual architecture of the FEDERICA infrastructure. During this activity, the need has been recognised to establish a new set of basic definitions (taxonomy) for the building blocks that compose the so-called slice, i.e. the virtual network instantiation (which is *virtual* with regard to the abstracted view made of the building blocks of the FEDERICA infrastructure) and its architectural plane representation. These definitions will be established as a common nomenclature for the FEDERICA project. Other important aspects when defining a new architecture are the user requirements. It is crucial that the resulting architecture fits the demands that users may have. Since this deliverable has been produced at the same time as the contact process with users, made by the project activities related to the Use Case definitions, JRA1 has proposed a set of basic Use Cases to be considered as starting point for its internal studies.

When researchers want to experiment with their developments, they need not only network resources on their slices, but also a slice of the processing resources. These processing slice resources are understood as virtual machine instances that users can use to make them behave as software routers or end nodes, on which to download the software protocols or applications they have produced and want to assess in a realistic environment. Hence, this deliverable also studies the APIs of several virtual machine management software products in order to identify which best suits FEDERICA's needs.

1.1.1 Acronyms and Abbreviations

ASON	Automatic Switched Optical Network
E-NNI	External Network Node Interface
Gbps	Gigabit per second
GE / GbE	Gigabit Ethernet
GFP	Generic Framing Procedure
GMPLS	Generalised Multi Protocol Label Switching
GRE	Generic Router Encapsulation
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPPM	Internet Protocol Performance Metrics
IPv4, IPv6	Internet Protocol version x
ITU-T	International Telecommunication Union-Telecommunication Standardization Sector
KB/MB/GB/TB	Kilo/Mega/Giga/Tera-Byte
LCAS	Link Capacity Adjustment Scheme
LSP	Label Switched Path
MAC	Medium Access Control
mp	Multi-Point
MPLS	Multi-Protocol Label-Switching
NNI	Network Node Interface (aka Network to Network Interface)
NREN	National Research and Education Network
OIF	Optical Internetworking Forum
OSPF	Open Shortest Path First
OTH	Optical Transport Hierarchy
OTN	Optical Transport Network
OXC	Optical Cross Connect
POS	Packet Over SONET/SDH
pt	Point
QoS	Quality of Service
RFC	Request for Comments
RSVP	Resource Reservation Protocol
RSVP-TE	Resource Reservation Protocol – Traffic Engineering extension
SDH	Synchronous Digital Hierarchy
SFP	Small Form-factor Pluggable
SLA	Service Level Agreement
TCP	Transmission Control Protocol
TE	Traffic Engineering
UDP	User Datagram Protocol
UNI	User Network Interface
VC	Virtual Container
VCAT	Virtual Concatenation

VLAN	Virtual LAN
WDM	Wavelength Division Multiplexing

1.2 Document overview

This document is structured as follows:

First, the basic definitions and virtualization architecture of the building blocks that compose a FEDERICA slice are introduced. This taxonomy aims to be a common nomenclature not only for TJRA1, but for the whole FEDERICA project. The next topic covered in the deliverable relates to the description of the Use Cases that can be identified from the point of view of the user/researcher and infrastructure. There is a risk that this list of requirements can become too extensive and ambitious, and it has therefore been limited to the capabilities that the physical FEDERICA infrastructure offers. These encompass the set of functionality requests that FEDERICA has to support in order to make its resources available to the research community, and therefore provide users with a logical view of its abstracted physical infrastructure (the “slice”). A slice is a set of resources from the infrastructure that has been partitioned and virtualized by the physical infrastructure manager system. This slice is perceived by the user as equivalent to a real physical network under his/her domain; it maps to a logical partition of the physical resources. This process is understood as the infrastructure virtualization process.

Once the building block architecture has been described, the different tools and frameworks currently available (worldwide) are studied. This study is presented in Annex I and identifies the functionalities of those systems which can be used within the FEDERICA approach in order to dynamically build a slice, to evolve the manual process envisaged for the first period of the FEDERICA project. This study also includes an analysis of the extensions to the tools or frameworks that can be developed in order to cover gaps or missing functionalities that are required by FEDERICA.

In the last section, VMware and XEN tools for the management of virtual machines are studied. This study is focused on their APIs and SOA interfaces to manage VMware and XEN.

2 Basic Definitions and Virtualization Building Blocks

The main objective of this section is to provide a set of basic definitions in order to have a common nomenclature for the FEDERICA project. The term "Virtualization of Building Blocks" is introduced to classify what we call the Virtual Resources Architecture. Furthermore, in the sub-section called "Structure of the Taxonomy", some important aspects that have to be taken into account when dealing with virtualization are identified.

2.1 FEDERICA's Virtual Architecture Taxonomy

This section provides a set of basic definitions considered within the virtualization scope of FEDERICA. This nomenclature will be used for the process of creating a slice, and is the basis of the Virtual Network infrastructure. As previously commented, this nomenclature will be used not only within JRA1, but also for the users and the FEDERICA NOC operators.

- **Physical Resource** (PR): single network element or computing component such as a fiber, a router, a switch, a link, a server.
- **Physical Infrastructure** (PI): set of Physical Resources (PRs) that together compose a communication network.
- **Physical Domain** (PD): an administrative domain in charge of the management and operation of a PI. Within the project, FEDERICA will represent one PD.
- **Virtual Resource** (VR): an abstraction or logical instance to a PR which appears to function as a physical resource from the user point of view. A VR can represent a partition of a physical resource ("slice") or a collection of PRs ("cluster"). FEDERICA focuses on slicing composition.
- **Virtual Infrastructure** (VI): a set of VRs offered to a customer with selected control capabilities, also called a FEDERICA Slice. The customer can use a VI to provide a Virtual Network Service (VNS) to other parties (end-users), or could even redistribute directly some of these VRs to third party. A VI can be used to provide the same kind of services as a PI, since it could also include the provision of a VI (recursive approach).
- **Virtual Network Service** (VNS): a service offered to end-user, resulting of a combination of VRs from one or multiple stakeholders, with specific SLS/SLA. Examples could be: L2VPN, L3VPN, BoD, Premium IP, etc. These services would require specific engineering and support from provider(s) and are offered as a part of a catalogue of services. The end-user has access to the service via a well defined interface without direct control of the PRs/VRs used for that service.
- **Virtual Infrastructure Service** (VIS): a service consisting of offering a VI to customer. There may be different forms of VIs (e.g. pure connectivity, IP infrastructure, computer facilities, etc.). Each service is defined by a VIS template, which should be instantiated into a VI.

- **Service Plane** (SP): the role of this plane is the orchestration of VRs to create a VIS template, and to create a VI (VIS instance) from a VIS template. The envisaged technology used for this plane is SOA/Webservices. The SP will be used to represent any PR as a VR (as a software object) in order to get an abstraction of them. The SP interacts with the Management Plane (MP) that controls the resources, in order to activate/deactivate/monitor VRs related to a VI.
  - The FEDERICA SP is located in the PI, and provides a VIS to FEDERICA's users.
  - Within the context of a user's VI, a SP should be implemented in order to recursively provide some other VIS.
- **Management Plane** (MP): it is a protocol for controlling the PRs and subsequently the VRs. It can have many forms, such as CLI/SSH, XML/netconf and SNMP.
- **Control Plane** (CP): the role of this plane is coordination between PRs and/or VRs to enable and sustain (monitor, reliability) a service ordered by the SP. A Control Plane may work intra- or inter-domain, and include many forms such as nothing, IGP discovery, BGP discovery and signalling and RSVP signalling.
- **Data Plane** (DP): it is in charge of handling transfer and treatment of information in the physical and virtual infrastructure.
- Note that each plane is present in the PI, as well as in each VI, except the FEDERICA SP which is present in the PI. A SP can also be implemented within a VI. Such a SP would be completely separate from, and independent of, the FEDERICA SP.

The following picture provides a pictorial description of the virtual FEDERICA infrastructure, including all the elements that have been previously described.

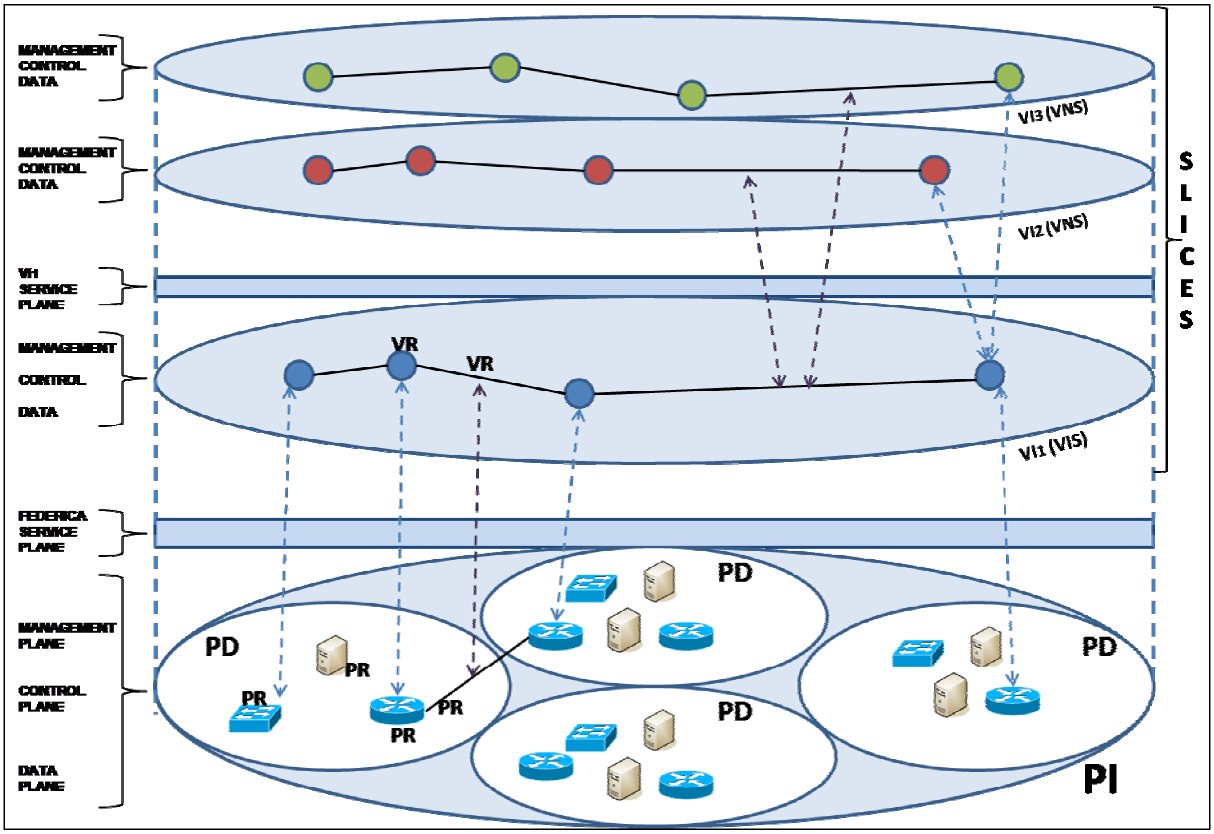

Figure 1: View of the FEDERICA Infrastructure

2.2 Virtualization Building Blocks

This section presents the key building blocks that can be virtualized and defined within the context of the FEDERICA network. These building blocks are classified in three types:

- EndPoints
- Connectivity
- Processing

Therefore, this section provides a mode detailed description of the FEDERICA's VR, focusing on the ones that are the main VRs in FEDERICA during the first year.

2.2.1 EndPoints

An EndPoint can be defined as a computer that consumes or offers a service (video streaming, transfer files, videoconferencing), so it is a machine that sends or receives data information through a VI. In other words, an EndPoint is a client or server that terminates a transport connection, such as TCP/UDP connection, more generally that consumes or feeds the network service.

An EndPoint has several characteristics provided as a virtualized instance. These are (for example) the Operating System modularity and the segmentation and the Ethernet card capabilities. This means that the Ethernet card is a property of an Endpoint, but not an Endpoint by itself.

Some of the technologies that allow this type of virtualization of computer resources are VMWare, XEN, User Mode Linux (UML), QEMU, Virtual Box, Virtual Iron, Linux KVM, Linux Vserver and Open VZ. These technologies will be explained later. From now on, the combination of these virtual machines together with the software installed on each of them will be called **Virtual Nodes (V-Nodes)**.

A virtual machine is one of the main components, together with the Logical Routers (VRs) that will be used in FEDERICA. We define it as a computing resource, typically realized through virtualization software (VMware and XEN in FEDERICA) running on a computer. Its main characteristics in terms of capabilities and functionalities are:

- Shared or exclusive access to parts of the underlying hardware. Network interface hardware can be shared between several virtual machines, for instance, or it may be assigned exclusively to a virtual machine.
- Limits on Operating System resources, such as disk space, CPU and memory assigned to the virtual machine.
- Restrictions on the Operating System. The underlying virtualisation software may impose restrictions on the Operating System in the virtual machine. (XEN para-virtualization would preclude Windows in the virtual machine for instance.)
- Restrictions on software. Certain software may not be able to run on the virtual machine. In particular, there are several limitations related to running virtualization software on a virtual machine.

2.2.2 Connectivity

Connectivity refers to a physical element of infrastructure that can be virtualized to offer information transfer within a VI (also called a "slice"), either point-to-point, or point-to-multipoint. Different technologies are envisaged to be used in order to accomplish virtualization in the different layers.

For example, SDH and OTN allow the virtualization of Layer 1 circuits, while VLANs allow virtualized services at Layer 2. Ethernet is a family of technologies for LANs. In order to virtualize Ethernet connectivity, some of these technologies such as VLANs, [IEEE 802.1q] Q-in-Q [IEEE 802.1ad], MAC-in-MAC [IEEE 802.1ah] are used. For example, VLANs virtualize LANs and more specifically the broadcast Data Plane and in some cases the Forwarding Plane (spanning tree), Q-in-Q virtualizes Ethernet trunks (dot1Q) and MAC-in-MAC virtualizes MAC functionality in the sense that it is possible to embeds Medium Access Control addresses in another MAC. Finally, MPLS VPNs allow L2 and L3 virtualization.

From the FEDERICA point of view, we can consider a virtual link as one Virtual Resources. It can consist of a VLAN or an MPLS LSP interconnecting different VR within a VI, and its main characteristics include the following aspects:

- Supported protocols and payload formats. VLAN tagging by the user may or may not be supported depending on whether Q-in-Q is available in the PR, for example.
- Traffic characteristics and service. Statistical multiplexing, traffic management, …

2.2.3 Traffic Processing

By the term "Traffic Processing", we refer to functionalities offered by devices that perform "intelligent" treatment on packets and/or flows of packets, e.g. dynamic routing, deep packet inspections.

Definitions of new virtual elements used when virtualizing devices that perform traffic processing are introduced below:

- **Virtual Router**: virtualizes the Routing Table of a router. It is a type of simplified routing instance that has a single routing table.
- **Logical Router**: virtualizes a Physical Router. It is a partition of a Physical Router and can contain multiple routing instances and Routing Tables. For example, a Logical Router can contain multiple Virtual Router routing instances. Some large manufacturers, like Juniper and Cisco, are currently providing software-based and hardware-based Logical Router implementations
- **Virtual Switch**: a partition of a Physical Switch that behaves like a Physical Switch.
- **Virtual Firewall**: a slice of a Physical Firewall device, with its own address book, policies, and management (Juniper).
- **VPN SSL Virtual Systems**: same as Virtual Firewalls but for a VPN SSL (this is implemented for the Juniper boxes that compose the FEDERICA PI).
- **Virtual Node dedicated to traffic processing**: it can be a slice of a Computing Element with either routing software or switching software.

2.3 Structure of the Taxonomy

Regarding the "virtualization of building blocks", several aspects were taken into account when evaluating the systems. First, we studied virtualization in three different planes: Data Plane, Control Plane and Management Plane. Thus, it was necessary to clearly define the set of capabilities and functionalities between the PR and the VR (slice) in all three planes. Another important aspect to explore was the capability of running different functionalities on each virtual slice, and finally, we explored security aspects; this study was orthogonal to the Data, Control and Management Planes.

3 FEDERICA Use Cases

The aim of this section is to provide a list of Use Cases from two points of view: (i) the user/researcher and (ii) the FEDERICA NOC infrastructure. This is presented as an initial list of Use Cases, which will be extended in future deliverables of JRA1 and later implemented as a service by SA2.

This deliverable presents a high level overview of a set of these Use Cases. A more detailed definition of their functionalities and implementations will be defined in future work (in both Activities, JRA1 and SA2).

These Use Cases have already considered the technical limitations of the physical network infrastructure from FEDERICA, which initially only considers L2 and L3 services, but not L1. Therefore, virtualization can only be performed at those levels. As a consequence, the Use Cases are focused on real scenarios that could become real implementations inside the project.

The technical limitations are related with the following aspects. First, Q-in-Q can only accept two levels of VLANs, so it is not possible to permit recursive slicing of interfaces. The first level will be used by the NOC to create the topology that a user requests. The second level will be used by the user to configure the VLANs needed to perform the experimentation.

Second, related with the creation of logical routers/switches, Juniper devices only accept one level of slicing. This is used by the NOC to create logical routers. The user gets a logical router allocated to him by the NOC. This logical router appears to the user as a normal physical router, with all the capabilities of a physical router but without the functionality of being able to create further logical routers from it. It is impossible to permit recursive slicing because the infrastructure does not accept more than one level of slicing.

Finally, with regard to the VLANs, the user can create as many VLANs as necessary in a slice (not Q-in-Q) and can use any VLAN id, because they are created above the VLAN preconfigured by the NOC to create its topology. Consequently, only the end-user slice reads the private VLAN ids that the user defines. Note that this makes the monitoring of the whole network more difficult because not all the devices are controlled by the NOC (only the initial FEDERICA configuration).

3.1 Users/Researchers Use Cases

In this section we explain the different Use Cases from the user/researcher point of view. These Use Cases are defined in terms of what the user may want to do, how the requests should be communicated to the FEDERICA NOC and what the user can do with the obtained resources.

3.1.1 Use Case 1: L2 Experiments

In this Use Case, a researcher wants to perform some L2 experiments.

Thus, he/she requests a L2 network composed of Ethernet switching devices.

As a consequence, the user obtains an infrastructure over which he/she has privileges for configuring different parameters of the VR of his/her slice. The goal of the user/researcher is to experiment with mechanisms and protocols allocating and managing virtualized resources at L2 (create VLANs (inside a VLAN or Q-in-Q), configure the Spanning Tree Protocol, create an instance of MST, configure OAM, etc). This involves experiments which, in general, need to manage the Data Plane, Control Plane and Management Plane at L2.

3.1.2 Use Case 2: Routing Management

In this Use Case, a researcher wants to:

- Test his/her IP management applications with an slice composed of an IP network VR.
- Configure routing (included external peering) parameters for experimental tests with standard routing protocols. (i.e. calculating convergence time of the existent and configured protocol when a node fails).

Thus, the user requests a dedicated IP network composed of a set of routers with higher performance (Logical Routers) or software routers (built on Virtual Machines) with lower performance but higher configuration capabilities, and also access links and backbone links. This infrastructure is requested with a specific topology and optionally indicating some QoS requirements.

Finally, the researcher obtains a slice composed of an infrastructure over which he/she has the control to change the IP configuration.

3.1.3 Use Case 3: Applications

In this Use Case a researcher wants to experiment (or to allow third users to experiment) with some applications that need to run on a distributed environment or imply some networking capabilities.

The User can make two types of experiments. First, he/she can test his/her applications without changing the IP configuration of the network. On the other hand, he/she can test his/her applications using different configurations of the networking devices of his/her network (for instance, test his/her application performance depending on the routing protocol used).

Thus, he/she requests a dedicated L2/L3 network composed of hosts (choosing the OS from a specific list) where he/she can deploy the preferred applications, and Network Elements, i.e., routers (Logical Routers and also software routers) and switches, with a specific topology.

At the end, the researcher has an infrastructure with full control of his/her hosts or maybe specific (from non to full control) permission on the routers configuration depending on his/her profile.

3.1.4 Use Case 4: Operating System related experiments

In this Use Case, an experimented user/researcher wants to:

- Test his/her applications (or allow third users to test their applications).
- Install switching or routing software (developed by him or by other entities) that runs over the Operating System of the Virtual Host for testing these routing/switching features (i.e. a new routing protocol).

In this way, the user requests an infrastructure composed of Virtual Hosts (choosing the Operating System from a specific list of them). In this case, the Network Elements (the topology) used to connect Computing Elements (where the Virtual Hosts are located) are transparent for the researcher and he/she can not change its configuration.

Then, this user obtains an infrastructure and can configure any application on the Virtual Hosts that belong to his/her slice.

3.2 FEDERICA NOC infrastructure Use Cases

In this the section different Use Cases are explained from the infrastructure point of view (FEDERICA NOC System).

The goal is to determine the way the system acts over the infrastructure to provide the different Use Cases defined by the end users/researchers. Thus, besides the description of the Use Cases, a mapping is included between the infrastructure point of view and the users/researchers point of view.

In all cases, there are two kinds of users that participate in the Use Case. The operator, i.e. the NOC administrator, who offers the services to build the Use Case (receives the request or the user); and the end user/researcher who asks for the Use Case (sends the request to the NOC administrator).

In general, the operator configures different resources in the FEDERICA infrastructure in order to sub-allocate them to an end user/researcher. The resources are organized in such a way that it allows for further virtual resource management under control by the end user/researcher.

3.2.1 Use Case A: Virtual LAN (mp-mp L2 connectivity)

This Use Case is applied when users want to perform L2 experiments (Use Case 1 from the user/researcher point of view).

The goal of the end user/researcher is to experiment with mechanisms and protocols allocating and managing virtualized resources at L2. This involves experiments that, in general, need to manage the Data Plane, Control Plane and Management Plane at L2. This service can be offered by FEDERICA with Ethernet switching technology only (VLAN, Q-in-Q, virtual switch), or combined with MPLS VPLS in the core.

This service would be offered as a VI, which means giving some control of the VR to the user.

3.2.2 Use Case B: Virtual IP infrastructure

This Use Case is applied when users want to perform some L3 (IP) experiments (Use Case 2 from the user/researcher point of view).

The user requests a dedicated IP network, composed of a set of routers (Logical Routers), access links (if they are dedicated only a normal link, if they are shared they have to be VLANs) and backbone links (the same conditions apply as for access links). The user requests a specific topology with the possibility to ask for some QoS parameters (such as BW, CIR, CBS, PIR, PBS, delay, jitter, etc.).

The operator configures different L3 resources in the FEDERICA infrastructure in order to sub-lease them to an end user/researcher. The resources are organized in such a way that it allows for further virtual resource management under control by the end user/researcher.

The idea is to provide the maximum possible flexibility to the user; consequently the user has the available resources as physical routers and physical interfaces and can specify the topology he/she wants (restricted by physical connections). The user will provide to the NOC a list of virtual interfaces with some QoS parameters (optional) associated to physical interfaces and routers.

Once the NOC has sub allocated the IP infrastructure, then the researcher can:

- Test his/her IP management applications with this IP network slice as a testbed.
- Configure routing (included external peering) parameters for experimental tests with standard routing protocols. (i.e. calculating convergence time of the existent and configured protocol when a node fails).

3.2.3 Use Case C: Virtual Computer Facilities

This Use Case is applied when users want to test their applications and software over Virtual Hosts (Use Case 4 from the user/researcher point of view).

An experimental user can request an infrastructure composed of Virtual Hosts (choosing the Operating System from a list of them) in order to have the highest privileges for the network configuration. The user requires a set of Virtual Nodes connected together, in order to experiment the behaviour of new software on a real environment.

The operator configures the requested Virtual Nodes in the FEDERICA infrastructure in order to sub-lease them to the end user/researcher.

The Virtual Nodes are offered as VI, but connectivity between them is offered as a VNS. This Use Case is dedicated to researchers that want to concentrate on their software applications and avoid the complex management of the network that connects the nodes. The Network Elements used to connect the Computing Elements (where the Virtual Hosts are allocated) are transparent for the researcher and he/she cannot change their configuration.

3.2.4 Use Case D: Virtual Network Laboratory

This Use Case is applied when users want to experiment with applications that require some networking capabilities (Use Case 3 from the user/researcher point of view).

The user requests a L2/L3 network composed of hosts and Network Elements, such as routers (it can be Logical Routers and also software routers) and switches in order to connect these hosts with a specific topology.

The operator configures the requested resources in the FEDERICA infrastructure slice in order to sub-lease them to the end user/researcher.

Then, the user can reproduce his/her experiments or prepare the scenario for third parties, configuring the hosts with the configuration that the experiment requires. It is useful for testing applications using different configurations of the networking devices of his/her network.

4 FEDERICA Virtual Infrastructure Requirements

In this section we define a set of basic requirements for FEDERICA. This will be useful when designing the toolbench application to dynamically create the virtual slices. These requirements have been classified in the following groups:

- Virtual requirements
- Operational requirements
  - AAA requirements
- Control Plane requirements
  - Layer 2 requirements
  - Layer 2.5 (MPLS) requirements
  - Layer 3 requirements
  - Computing Element (PC) requirements
- End user requirements
- Monitoring requirements
- User friendly requirements

These basic requirements have been checked against all thetools/frameworks studied (see Annex I), in order to make a comparison of them and choose the main candidates to be used for the FEDERICA deployment.

Virtual requirements refer to the physical resource representation and partitioning functionality:

- Representation and control of physical resources
- Resource partitioning:
  - Switch slicing (creation of VLANs)
  - Router slicing (creation of logical routers)
  - Computing Element slicing (creation of VMs in a V-Node and make them work as a router/switch or install applications in them)

Operational requirements refer to the general characteristics of the infrastructure and the toolbench:

- Isolation between slices in a multi-user environment
- Inter-slice communication capability
- Interconnection with other infrastructures
- Connection to the Internet
- Possibility to apply changes to the network manually
- Possibility to modify the slice after the original request
- Reproducibility of experiments during lifetime of an experiment

AAA requirements are included in the operational requirements, and they are one of the most important components of the FEDERICA project:

- Provide a policy based access control
    - Allow user roles definition
        - Operator
        - End user (researcher)
- Provide global accounting system
    - Support business models (billing)
    - Services for collecting and publishing accounting data with the proper level of privacy and reliability
    - Trace and limit usage of resources

Control Plane requirements

The Control Plane of the toolbench will interact with all the layers, planes, technologies and domains so that end-to-end Virtual Network services may be offered. The basic Control Plane requirements are:

- Security: security at different layers
- Model stitching between different technologies
- Support stitching between different domains
- Allow requirements mentioned later in L2/L2.5/L3/Computing Element

Layer 2 requirements:

- Managing VLANs
    - Creating/Deleting VLANs
    - Enable/Disable VLANs
    - Displaying VLANs
- Adding static members to VLANs
    - assign VLAN-ID to a port
- Configuring VLAN behaviour for Interfaces (GVRP)
    - GARP VLAN Registration Protocol defines a way for switches to exchange VLAN information in order to register VLAN members on ports across the network.
- Configuring VLAN Trunks
    - A trunk is a point-to-point link between two switches that carries the traffic of multiple VLANs over a single link.
- Configuring Private VLANs
    - Private VLANs provide port-based security and isolation between ports within the assigned VLAN.
- Configuring Protocol-Based VLANs
    - Divide the physical network into logical VLAN groups for each required protocol
- Q-in-Q VLAN tagging
    - expand the VLAN space by tagging the tagged packet

    - allows service providers to insert an additional VLAN tag in the Ethernet frame in order to identify the service
- MAC-in-MAC scaling
    - encapsulates Ethernet frames with a MAC header
    - overcomes the inherent scalability limitations of VLAN and Q-in-Q networks tag in the Ethernet frame in order to identify the service

Layer 2.5 (MPLS) requirements:

- Creation of LSPs
- Application of QoS policies
- Creation of L2 VPNs
    - L2VPN (VLL)
    - VPLS (Multipoint L2VPN)
- Creation of L3 VPNs

Layer 3 requirements:

- Use of IPv4 and IPv6
- Request/Configure a specific topology
- Application of the different routing protocols
    - Current routing protocols
    - New routing protocols
- Allow testing new Layer 3 capabilities
    - Routing
    - Path discovery
    - Constraint based path computation

V-Node requirements:

- Use of IPv4 and IPv6
- Creation of different Virtual Machines (VMs) in a V-Node
- Resources allocation: memory and CPU consumption
- Allow installing a specific application in a VM working as a host
- Configure a software switch in a VM
- Configure a software router in a VM
- Possibility of managing VMs via Web-Services

End User requirements:

- Sub-allocate set of virtual resources to the end user with the control associated to its user profile (full control or limited control). This sub-leasing will be for a limited time period.
- Choose a specific topology for testing the new applications/protocols: create his/her own networks (at different layers) including in it a set of his/her own sub-leased resources

Monitoring requirements:

- Get statistics from the various devices (physical and virtual) included in the FEDERICA infrastructure
- Get notifications from the various devices, indicating if any errors have occurred
- Visualization of the current configuration of the infrastructure
- Allow user get monitoring information depending on the user profile

User friendly requirements:

- Software mechanism to represent the various devices (physical or virtual) and their configuration (if the tool has a data structure and a graphical interface for example)
- (User friendly) toolbench, having:
  - The facility to create queries to the system
  - Use of understandable terminology
  - Parallel configuration: configure a set of similar devices at the same time
  - Customisation at request: the flexibility offered by the tool to perform users' requests.

5 Available Tools/Frameworks

5.1 Introduction

This chapter introduces a set of available tools and frameworks that can be useful for the FEDERICA project. These tools and frameworks are already developed and deployed, and being used in research projects on a worldwide scale. JRA1 has made an exhaustive study and analysis, in order to see which ones fit best the FEDERICA project or could be extended in order to fit the FEDERICA requirements. This study has been made because FEDERICA does not intend to build a new tool from scratch, but rather take advantage of developments that have already been made by the research community. As the main objective is to evaluate available control and management architectures and protocols, which may be applicable to a multilayer infrastructure in a multi-domain Virtual Network environment, this comparison study is presented in Annex I. This chapter presents an overview of the tools analysed in Annex I.

5.2 Overview of Tools/Frameworks

The tools/frameworks studied in this activity can be divided in two general groups: (i) network provisioning tools and (ii) network monitoring tools. The IaaS Framework and the PL_VINI are located within the first group, and both can be used to provide virtualization services.

- Network provisioning tools: AMPS, ANStool, ARGON, AutoBAHN, BLUEnet, DRAC, DRAGON, MANTICORE/Argia & IaaS Framework, Juniper SRC, PL-VINI.
- Network monitoring tools: GINS, G3 System.

5.2.1 AMPS

AMPS (Advance Multi-domain System) was developed in the GÉANT 2 (GN2) EC-IST project, and enables authorized end-users to make a single reservation for Premium IP (PIP) bandwidth (i.e. a guaranteed uncongested path) that is effective along a chain of participating domains.

AMPS is service-oriented, meaning that the application contains several modules (or services) which may be deployed individually, or in combination, on one or more physical servers. Briefly, these services are the following:

- Inter-domain Module: exchanges messages with AMPS servers between neighboring (peer) domains, receiving and forwarding PIP requests, acceptances and rejections.
- Intra-domain Module: stores information about accepted PIP reservations, and decides whether new requests can be satisfied or not.
- Path-finding Module: called by the Intra-domain module in order to calculate the path an IP flow will take across a domain.
- Net Discovery Module: allows a quick and automatic dissemination of the AMPS network topology database.

- Policy Module: accepts or rejects user requests for PIPs, based on policies pre-defined by the network administrator.
- Config Module: Generates the configuration that needs to be applied to a router in order to support a PIP flow. In the future the module will allow the automatic configuration of the routers but currently the configurations are just prepared, ready to be "cut and pasted" by NOC engineers.
- GUI Client: not strictly part of AMPS, this client allows human users to submit PIP requests directly to AMPS, instead of using another application, e.g. EGEE middleware.

5.2.2 ANStool

ANStool (Advanced Network Services), built by GRNet, is a simple and extensible framework that allows to networking engineers to get information about the network, store this information in a database and provide - optionally - QoS reservation inside a Layer 2 or Layer 3 VPN. ANStool provides a Web-based application (PHP) for establishing unidirectional and bidirectional bandwidth reservations using various technologies within a single domain. It can also assess the feasibility of the request.

ANStool works mainly in the OAM Plane. It does not communicate directly with the Control Plane of the underlying network devices. The tool is able to manage Layer 2/3 devices (currently Cisco and SNMP managed devices) and is currently being modified to model also Layer 1 (WDM) devices. It has a centralized topology, mapping functionality with distributed control over the devices. The provisioning is done in a semi-automatic way. The configuration is produced automatically but the corresponding administrators must send the configuration to their own networking devices, until sufficient trust and security is guaranteed.

5.2.3 ARGON

ARGON (Allocation and Reservations in Grid-enabled Optical Networks) is a centralized system that manages and allocates network services on-demand and in-advance with QoS within a single administrative domain. It is specially focused to be used with Grid applications and Grid scheduling, however, one or more ARGON instances can be integrated easily into a multi-domain environment by means of the Harmony/NSP interfaces developed in the EC-IST Phosphorus project.

ARGON operates on different network layers (Layer 2 via GMPLS / VLANs / VPLS, Layer 3 via MPLS) and offers different service types for inter-cluster communication and data transfer. It also provides interfaces for the co-allocation of heterogeneous resources in the Grid. Two core services can be used:

- A pipe service that provides end-to-end connections with user specified QoS within a time frame.
- A malleable service that provides connection with a network determined QoS to support data transfer tasks.

Resources are manually assigned to ARGON. These resources are configured via SMNP and CLI, and are enriched with availability information over time, i.e. ARGON is aware of the bandwidth throughout time. The tool finds and establishes shortest paths given the constraints specified by the user. These constraints can be: time (start time, duration), bandwidth (minimum bandwidth, maximum bandwidth), and delay (maximum delay). It also supports a mixed time/bandwidth constraint, the amount of data to be transferred by a certain deadline, with the "malleable reservation" type.

ARGON supports multi-domain architectures and reservations. A reservation request can be comprised of several services (session concept), and therefore, it can be applied, as a whole, as an admission mechanism of a more complex communication structure.

5.2.4 AutoBAHN

AutoBAHN (Automated Bandwidth Allocation across Heterogeneous Networks) is a bandwidth reservation and signalling interface system that allows end-users to make advance reservations with automated provisioning between interconnected domains (peer-to-peer) that may be deployed over heterogeneous network technologies (L1 and L2).

AutoBAHN is a distributed system that requires a domain manager in each domain; within each domain the policies and implementations are independent. The domain manager is split into two modules, the Inter-Domain Manager, which is responsible for inter-domain operations of circuit reservation on behalf of a domain, and the Domain Manager which is the module responsible for instantiation of BoD instances within a single domain. It also contacts the Technology Proxy module to request the configuration of the BoD service instance. A Technology Proxy module, allows to AutoBAHN the support of a wide range of technologies and vendors according to multiple domain and global requirements.

Being a modular system allows the support of external services such as the authorization and authentication infrastructure (AAI) or other existing functionalities, for example the Intra-domain Manager. The system is based on a set of specific interfaces and supports unique implementations in individual domains.

An abstract network resource representation is used to describe the specific technological details in a common format. This provides a non-ambiguous, simple and common language to all the AutoBAHN's system components and, in particular, to negotiations between domains.

The AutoBAHN system is managed in a centralized way by a dedicated AutoBAHN GUI (Web portal) which gathers information about the accessibility of the domain managers that have been registered to the GUI via Web-Services. The GUI allows the request and cancellation of reservation services, as well as the representation of a map with registered domains. With this Web portal user can also monitor states of submitted reservations.

5.2.5 BLUEnet

The BLUEnet Configuration Tool is based on the ANStool developed by GRNET and modified to support MPLS L2 circuits. It has been developed and implemented by HEAnet in order to automate the provisioning of Ethernet point-to-point links over the HEAnet network (intra-domain). The BLUEnet tool performs semi auto-discovering and automatically maps new devices added to the network, as well as allowing users to select the interfaces (two end points) and configure the link via a Web-based interface.

The main use of this tool is the establishment of "Port mode service" and "VLAN mode service" connections in an MPLS network. It configures Port mode/VLAN mode links over native Ethernet and L2 MPLS VLL clouds.

- Port mode service: it is a point-to-point port-based transparent Ethernet virtual circuit that provides connectivity for both Layer 2 data and Control Plane data. It is used to connect geographically remote LANs over the HEAnet network. Customer 802.1q ports are configured to perform Q-in-Q.
- VLAN mode service: it is a point-to-point VLAN-based Ethernet virtual circuit. The service is non-transparent, and VLANs are mapped to an MPLS label switched path. The user sees an 802.1q trunk which filters customer Layer 2 control protocols.

The configuration of circuits is done using a Web-based GUI, while the creation and deletion of circuits uses Web-Services. Nagios and Cricket (two monitoring tools) are automatically created/ deleted for each new circuit.

5.2.6 DRAC

The main goal of the DRAC (Dynamic Resource Application Controller) is to allow applications to configure the network without requiring to interface directly to a wide range of diverse and constantly evolving network protocols, features and devices. DRAC works on an optical and packet switching architecture (L1 and L2) and it provides connectivity at Layer 3, but its architecture is composed of Layer 1 and 2 devices.

DRAC is focused on Grid applications. It aims to provide connection between different hosts used by a Grid application with some QoS parameters. Once this goal is achieved, the Grid application can access the host with a specified bandwidth, delay, jitter, packet loss, etc. The foundation that DRAC uses for the provisioning of these QoS parameters is “cut-through”; a switching method for packet switching systems, wherein the switch starts forwarding a frame (or packet) before the whole frame has been received, normally as soon as the destination address is processed. This technique reduces latency through the switch, but decreases reliability. DRAC uses this method and steers packets in Layer 1 instead of Layer 3. Doing that, it can achieve better QoS requirements, and as a result, a better network performance.

Briefly, the operation of the tool is as follows: applications ask to DRAC for a path between to end-points of the network with a specific QoS. Upon receiving such a request, the tool tries to find an available path across the optical and packet switching architecture. Once a path is found, it validates the QoS requirements, and if they are achieved it virtualizes the path, transforming it into a virtual switch. The two end-points of the path are the two interfaces of the switch, providing a transparent connectivity to the application. After the virtual switch is generated, DRAC validates the QoS requirements in the extremes of the virtual switch, for finally sub-leasing the instantiated path to the application.

5.2.7 DRAGON

DRAGON (Dynamic Resource Allocation via GMPLS Optical Network) is a network architecture that defines an improved Control Plane (Figure 2) to enable multi-domain, multi-layer and multi-service provisioning. The architecture consists of multiple autonomous network domains, where each domain can define internal traffic management policies, and enter into peering arrangements with other external domains. It is based on switching and forwarding nodes that support the GMPLS label hierarchy.

The main objective is inter-domain instantiation of end-to-end LSPs which can be dynamically provisioned across multiple administrative domains and heterogeneous network technologies, but there are several capabilities and technologies missing in order for this capability to be available to end systems yet:

- Standardized inter-domain routing architecture for LSPs.
- Simple application interfacing.
- End to end instantiation (with proper authentication, authorization, and accounting).
- The ability to signal across non-GMPLS enable network segments.

The main modules of the DRAGON are:

- Network-Aware Resource Broker (NARB): It represents the local Autonomous System (AS) or domain and serves as a path computation engine and inter-domain routing. NARBs peers across domains and exchange topology information based on the actual topology as discovered by listening to the local OSPF-TE protocol, or optionally based on an “abstracted” view of the domain topology. It also includes advanced algorithms which allow path computation with multiple constraints (GMPLS TE parameters as well as AAA, scheduling, multi-region switching capabilities, and vendor specific limitations such as switching capability adaptation abilities).
- Application-Specific Topology Builder (ASTB): Application-specific Topologies (ASTs) are requested by an end user and are generally a set of LSPs in which an application domain desires to be set up as a group. The ASTB accepts requests from users or end systems for multiple network connections, and uses the services of the NARB to determine if the requested network paths are available with appropriate AAA and schedule the applied constraints. The NARB views these requests as individual LSPs and the ASTB is responsible for the assembly of multiple LSPs into a specific topology.
- Virtual Label Switch Router (VLSR): A non-GMPLS capable network device (Ethernet, TDM and Optical switches) is converted to a VLSR by the addition of a small UNIX-based V-Node which runs a GMPLS Control Plane consisting of OSPF-TE and RSVP-TE. The VLSR V-Node acts as a GMPLS proxy agent for a device and translates protocol events into commands that the local switching element understands, such as SNMP, TL1, or even scripted CLI commands. This allows non-GMPLS devices to be included in end-to-end path instantiations.
- End-System Agent (ESA): it is software that runs on the end-system that terminates the Data Plane link of the provisioned service, and it allows the initiation of a provisioning action on behalf of the end system. It may also interact with the ASTB if a more complicated topology is to be built.

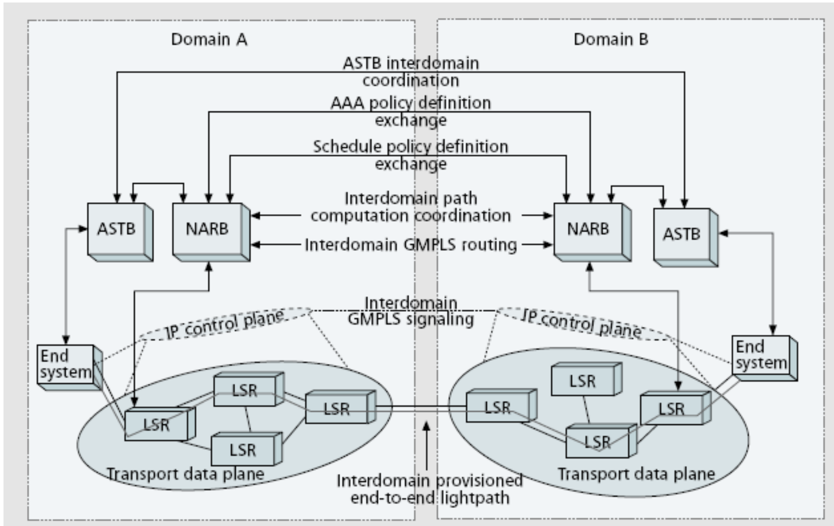

Figure 2: DRAGON Control Plane Architecture

5.2.8 G3

The G3 (Figure 3) System has been developed at CESNET and is designed to provide large scale and continuous network infrastructure monitoring. Measurement mechanisms and data processing are able to store and visualize some level of network dynamics. Data processing mechanisms ensure automated adaptability on real device reconfigurations and provide continuous and flexible mapping between the technological- (given by SNMP) and logical- (human point of view) structure of the measured devices. The user interface aggregates measured data while retrieving it from storage. Therefore the whole user interface is strictly designed as interactive.

The G3 System's interactive user interface is used by network administrators. It allows to define a time monitoring window ("from" to "now" basis), and the navigation is done through a tree representing devices and all SNMP measurable items for each device. The time stepping of measurements can be changed dynamically and there exists the possibility of viewing sums, minimums, maximums of aggregated values.

The reporter is an external standalone tool working above the interactive user interface which generates static reports retrieved by an HTTP server and accessible by end-users. The report content is given by specific configuration; a typical example is a report about CESNET2 IP/MPLS backbone utilization (link, capacity, bit rate, utilization). The data shown by the reporter is set by the network administrators using the User Interface and defining Saved Sessions. Also, a configuration file has to be administered in order to define how the data will be reported.

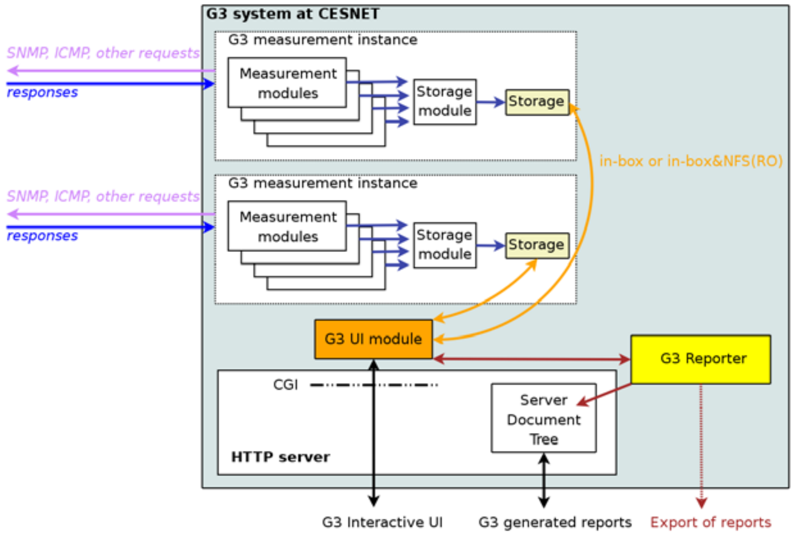

Figure 3: G3 detailed scheme

5.2.9 GINS

GINS (GARR Integrated Networking Suite) is a framework that integrates network monitoring tools, statistics acquisition tools, trouble ticket system, fault and performance reports using a user friendly GUI. The main objectives of GINS are the integration of all the monitoring tools already available and to be developed, and the provisioning of a dynamic tool configuration, on the basis of the information stored in the GARR database (GARR DB) and those obtained directly from the network equipment.

GINS has a centralized management and it is able to account intra-domain monitoring and to support inter-domain monitoring compliant with the perfSONAR framework of GÉANT. The main functionalities of GINS are:

- Monitoring functionalities: IPv4 and IPv6 interfaces, IP multicast beacons, routing protocols (OSPF, BGP), SONET/SDH routing interfaces, Lambdas status, MPLS status and E2E status (stitching of multiple links).
- Statistics functionalities: GINS stores performance measurements data and provides graphics throughout time about multiple metrics, such as: IPv4 and IPv6 traffic statistics, SONET/SDH error statistics, router CPU load statistics, end-user uncompressed statistics, etc.
- Fault and performance reports functionalities: GINS provides reports in HTML and PDF formats, some examples of the reports are the User Monthly Report and Carrier Fault Report and circuit availability.
- Data export functionalities: XML data can be exported to support perfSONAR and the GN2 E2E monitoring service

5.2.10 MANTICORE/Argia & IaaS

The IaaS (Infrastructure as a Service) Framework is an open-source framework descendant of the UCLP Research Program and made available by Inocybe Technologies and partners to allow people to create their own compatible middleware solutions. The framework is a set of software tools, libraries, and applications as well as documentation and best practices to follow in developing P2V (Physical to Virtual) solutions.

The framework manages devices from different domains allowing their interconnection. Each type of resource (switch, router, optical device) is managed by a Web-Service and each device is represented as a Resource (using WSRF). These Web-Services can be distributed in several machines.

The IaaS Framework offers common functionalities that can be used by several types of networks and devices. MANTICORE and Argia are services that use this framework, the former provides virtualization mechanisms for IP networks and the latter provides virtualization mechanisms for optical networks. Services can be activated though the GUI or with an application that calls the WS API.

The main objectives of this framework are:

- To enable IT infrastructure to follow business requirements.
- To anable no over-provisioning of IT infrastructure when estimating future needs, which permits using resources on an “as-needed” basis.
- Using virtualization, administrators can manage from 5 to 10 times the number of devices they are managing at the moment.
- The creation of virtual resources to partition networks into multiple sub-networks.
- To make available new resource development quickly and easily.
- To provide a uniform development environment.
- To provide interoperability across different middleware implementations. Virtualized resources are software resources that may be accessed by applications or other Grid Services.
- To provide common resources for different technologies.
- To provide reusable components which can be plugged seamlessly into new resources.
- To provide guidelines for best practices development.
- To allow a transparent migration to new features.
- To enable the use of Virtual Resources within a single domain or across multiple, independently managed domains.

5.2.11 Juniper SRC

The Juniper Networks Session and Resource Control (Juniper SRC) Portfolio is a carrier grade policy and control solution that integrates third party platforms and applications that enable the end-to-end delivery of high value differentiated services across multi-vendor network infrastructures. It consists of advanced hardware and sophisticated software that extends Juniper's experience in providing highly intelligent reliable and scalable IP network layer solutions to the policy and control layer. Key policy and control layer functions include policy management, subscriber management and authentication, authorization and accounting (AAA), and network resource control.

The Juniper SRC Portfolio supports a myriad of services, as for example:

- Multiplay Services (e.g. Bandwidth on Demand).
- Tiered Access Services and Subscriber Self Provisioning.
- Enhanced Security Services.

SRC supports the Policy and Control Management Plane, and it is connected to the Service Plane and the Transport Plane. The focus of SRC is to control the device of the operator domain, translating a service request into a policy in the Transport/Control Planes. For example, SRC supposes all the inter-domain coordination is performed at the Service Plane Layer.

The global functionalities of SRC are:

- Intelligent Admission Control: makes per-session admission control decisions.
- Real-time Bandwidth Management: enables allocation and reservation of network resources and bandwidth end-to-end across the access, edge, and core network.
- Dynamic Bandwidth Adjustment: policy-based dynamic traffic engineering mechanisms create and resize MPLS LSPs in response to dynamic subscriber and application requests.
- Synchronized Accounting: tracks and accounts for dynamically initiated services.
- Open Interfaces: Supports the integration of third-party network elements and applications to enable end-to-end, application driven network resource control.

The provisioning of services, i.e. to interface with the configuration of SRC, can be accomplished via CLI, a Web interface or NETCONF (XML). The configuration of the network equipment is done by SRC via COPS, RADIUS CoA, PCMM, XML and script services API.

5.2.12 PL-VINI

PL-VINI (Figure 4) is an evolution of PlanetLab targeted on constructing the network topologies using currently available routing and forwarding software (i.e. Click modular router, XORP router Control Plane). The main objectives of PL-VINI are: (i) building on-demand topologies as an overlay on top of the current infrastructure and (ii) enabling a high degree of customization using standard networking software.

The tool works as in a single domain but with resources distributed over the Internet. The control is centralized and the provisioning is automatic across the various PL-VINI nodes. Each of these nodes is able to host several slices. PL-VINI manages resource partitions using Vserver virtualization software and the VNET socket for network isolation between slices.

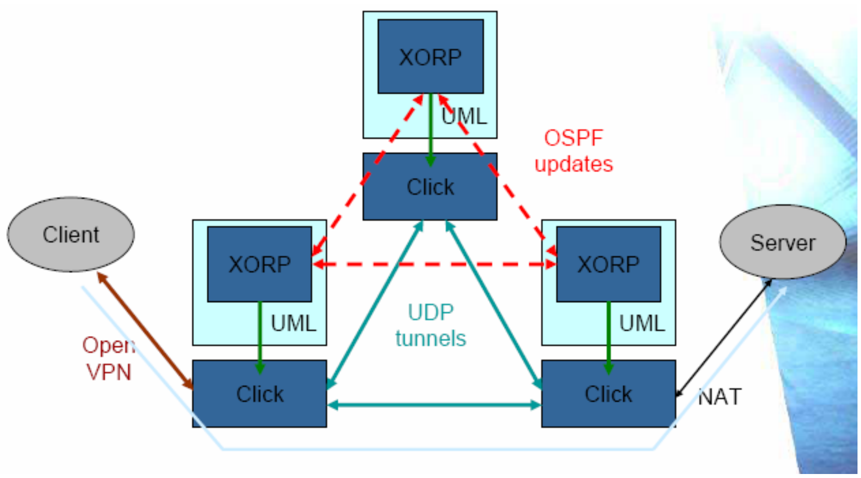

Figure 4: Internet in a Slice

6 Desirable/usable control and management functionalities and protocols for the FEDERICA network slice provisioning

The main objective of this section is the selection of a set of tools and frameworks that can be initial candidates for the process of virtualization and slicing creation of the FEDERICA toolbench. The outcomes presented in this section are derived from the comparative studies shown in Annex I. This study has been exhaustive as far as the documentation was available. The tools and framework analyzed in Annex I are prototypes or operational services/systems that have been developed by the research community, from a worldwide perspective. The motivation for this study was because currently there is no system capable to satisfy the FEDERICA needs. Therefore the objective is identify and match what tools can be reused and adapted to fit the complete FEDERICA requirements. After the table, there is a more detailed description of the arguments used to build it, and the implications that each tool and framework has on the FEDERICA requirements (from section 4).

6.1 Tools and Frameworks Comparison table

The table (Table 1) presented below is the result of the comparison study done for the various tools and frameworks presented at the Annex I. It aims to identify which are the tools and frameworks that best fit the FEDERICA requirements. The set of selected tools and frameworks can be considered as candidates that will subsequently be studied in depth within JRA1. However, although tools have been matched to certain requirements, it can be seen that only a combination of selected ones can fully fulfil them all.

It is important to note that this study will continue during the project, since some tools and frameworks may be improved, or even a new system may appear, due to the developments that other research projects have on the virtualization topic.

FEDERICA requirements \ Tools and Frameworks	AMPS	ANStool	ARGON	AutoBAHN	BLUEnet Tool	DRAC	DRAGON	G3	GINS	MANTICORE /Argia	Juniper SRC	PL-VINI
Operational										X		X
End User						X				X		
User friendly				X		X				X		
Control Plane		X	X	X	X		X			X		
Management Plane		X			X	X	X					X
Monitoring					X			X	X			
Virtual						X				X		
V-Node												X
AAA	X				X						X	X
Data Plane		X		X			X			X	X	
Layer 2			X		X							
Layer 2.5							X					
Layer 3	X				X		X			X	X	X
Licensing schema	X	X	X	X	X		X	X		X		X
Service Plane			X	X		X	X	X	X	X	X	
	3	4	4	5	7	5	7	3	2	9	4	6

Table 1: Comparison table of tools and frameworks

The conclusion raised from the Table 1 is that MANTICORE, BLUEnet, DRAGON and Pl-VINI are the best candidates for the tools or frameworks to be used within the FEDERICA project. These are the tools that provide features considered to be used (or enhanced) for the toolbench development, to provide the dynamic/ manual virtual slice creation functionality.

6.2 Operational functionalities

IaaS Framework and PL-VINI allow slice-isolation, inter-slice communication and interconnection with other infrastructures and the Internet. These functionalities are considered as important ones in the FEDERICA project and the toolbench will have to provide them.

PL-VINI provides complete isolation through Vserver, inter-slice communication via UML switch, interconnection with other infrastructures by using *openvpn* and connection to the Internet through NAT.

In the IaaS Framework services, each user has his/her resources blocked to the others, external peering can be configured to allow inter-slice communication, resources can be exported to be used in other infrastructure and vice-versa, and also IP parameters can be configured to go out to the Internet.

DRAGON and ARGON allow interconnection with other infrastructures; DRAGON via peering and ARGON using the solution of Phosphorus Harmony/NSP. AutoBAHN also allows interconnection with other infrastructures and the Internet.

Although most of the tools studied fulfil some of the FEDERICA operational requirements, the most complete tools are IaaS Framework and PL-VINI, especially in terms of the functions related to the slices (isolation, inter-slice communication, interconnection with other infrastructures and the capability of modifying a slice after the original request).

6.3 End user functionalities

The main end user functionalities are sub-leasing virtual resources to end users, and creating own topologies.

IaaS Framework services allow the exporting and importing of resources. This is similar to sub-leasing, because with these mechanisms other users have permissions over a set of virtualized resources. Despite this, DRAC can create a virtual switch with QoS parameters to sub-lease a point-to-point path to the end user.

DRAGON allows end-to-end provisioning with a chosen topology using the Application-Specific Topology Builder (ASTB) Module. The IaaS Framework (through Argia and MANTICORE) allows the creation of specific topologies by configuring interfaces and links. DRAC is also capable of creating its own networks. PL-VINI and ARGON allow own networks creation, but cannot sub-lease virtual resources to an end user.

As a conclusion, the IaaS Framework and DRAC would fulfil both end user requirements. The disadvantages are that DRAC and Argia are private products, and MANTICORE is at the end of its development stage.

6.4 User friendly functionalities

Most of the tools include some functionalities to offer a user friendly environment, such as software to represent devices, a query facility, etc.

A GUI is included in tools like DRAC and IaaS Framework that simplifies the queries, or representation of the devices. Many of them also work with an understandable terminology, as ARGON, AMPS, ANStool, BLUEnet, etc. Flexibility to perform users' requests is also fulfilled by many of the tools.

Comparing, we have observed that the most user-friendly tools are AutoBAHN and IaaS Framework. Also DRAC would be a good option because of its graphical interface.

6.5 Control Plane functionalities

Some important Control Plane functionalities for the FEDERICA toolbench that are implemented in some of the studied tools/frameworks are: (i) stitching between technologies, and (ii) stitching between domains.

Most of the tools/frameworks implement stitching functionalities between different technologies; these tools are ANStool, AutoBAHN, BLUEnet, DRAGON and Argia.

The BLUEnet tool provides links over native Ethernet and L2 MPLS VLL clouds.

AutoBAHN and DRAGON implement both technology and domain stitching, AutoBAHN allows configuring and interoperates with various L1 and L2 technology-specific domains (lambdas, MPLS VPNs, Ethernet, SDH and GFP over SDH). DRAGON implements the NARB (see sub-section 5.2.7) that provides functions to enable network provisioning on an inter-domain basis across topologies which include a heterogeneous mix of network technologies and vendor equipment. More specifically, these technologies are switching components that have their own native GMPLS protocols.

Summarizing, there are three tools that fulfil all the Control Plane requirements, AutoBAHN, DRAGON and IaaS Framework (through Argia and MANTICORE). ARGON and DRAC discover topology and route automatically. Moreover, ARGON also is capable to switch between domains. A combination of the BLUEnet tool or ANStool with ARGON would fulfil all the requirements.

6.6 Management Plane functionalities

The main requirements related to the Management Plane are resource discovery (done manually or automatically), fault recovery and security.

Automatic resource discovery is done automatically by ANStool (through GRNET DBII), the BLUEnet tool (using *Perl* scripts), DRAGON and PL-VINI.

Security management is offered by DRAC (using an external AAA), and PL-VINI (made possible via SSH).

None of the tools offers fault recovery management.

The most complete of the tools, for the Management Plane requirements is PL-VINI.

6.7 Monitoring functionalities

The monitoring requirements relate to obtaining statistics and notifications from devices, visualizing the current infrastructure configuration, and getting monitoring information.

GINS offers statistics for all resources with SNMP support (physical or virtual). G3 also offers statistics from devices. The BLUEnet tool shows link utilization graphs (though Cricket). PL-VINI also can obtain statistics from devices. GINS, G3, ANStool, AutoBAHN, IaaS Framework can visualize infrastructure configuration. BLUEnet can list all VPNs and see the device configuration.

GINS gets notification from devices by means of SNMP polls triggered by a centralized manager,

The BLUEnet tool and IaaS Framework also generate alarms. The Juniper SRC uses a polling done by SRC, based on deactivation or interim.

ANStool, the BLUEnet tool, G3 and GINS can get different monitoring information depending on the user profile. Concretely, BLUEnet shows link utilization graphs.

GINS offers fault detection and notification and supports monitoring in both private and public networks. It also can support slice changes. It supports Ethernet, MPLS, IPv4 and IPv6 monitoring as well as BGP and OSFP monitoring is supported in terms of routing protocols. It can also monitor different slices. GINS supports VLAN statistics.

Physical devices that are monitored by G3 are L2 and L3 switches and routers, SNMP capable devices, but they have to be discovered manually. GINS monitors equipment with SNMP MIBs (either physical or virtual). They also have to be discovered manually.

In terms of user-friendly monitoring, in G3 each device can be seen as a tree structure of objects to measure. GINS includes different DBs, and provides a user-friendly GUI to show all the collected info with different details and views.

Summarizing, although BLUEnet offers various monitoring options, it is not as complete as a dedicated monitoring tool would be. GINS and G3 are monitoring tools, and both of them are complete. The other tools offer only a few monitoring options.

6.8 Virtual functionalities

The most important requirements are the representation and control of PRs, and the resource partitioning or slicing capability. Two tools fulfil the requirements: (i) DRAC and (ii) IaaS Framework.

DRAC presents a good representation, but not full control, while IaaS Framework, through MANTICORE and Argia offers more control options. In terms of resource slicing, DRAC does not offer the possibility to create slices, but users can share the resources. On the other hand, IaaS Framework (using MANTICORE and Argia) allows in a more complete way to create slices by virtualization (creating logical routers, logical ports, etc.). As said before, DRAC and Argia are therefore tools that must be considered.

6.9 V-Node functionalities

PL-VINI fulfils almost every requirement. It is able to create virtual machines, allocate memory and CPU. It is also able to install applications in a VM and offers switching software and routing software (also offered by AutoBAHN). None of the tools allow managing a VM via Web-Services.

6.10 AAA functionalities

Many of the tools offer a policy-based access control, and some of them are able to trace and limit the usage of the resources. However, a “business model” is only supported by AMPS, IaaS Framework and the Juniper SRC. Only the BLUEnet tool, Juniper SRC and PL-VINI are able to collect and publish accounting data.

Summarizing, the Juniper SRC is able to fulfil all the AAA requirements. AMPS, the BLUEnet tool, IaaS Framework and DRAGON offer most of the functionalities. A combination of two of them could cover all the requirements (e.g. AMPS and the BLUEnet tool).

6.11 Data Plane functionalities

There are three important Data Plane requirements: (i) the supported technologies, (ii) the physical devices and (iii) the network connectivity.

ANStool and DRAGON support Ethernet, MPLS and IP. AutoBAHN and the BLUEnet tool support Ethernet and MPLS. The BLUEnet tool also supports VLANs. DRAC supports only Ethernet, and IaaS Framework and the Juniper SRC only support IP. AMPS supports only IP Premium.

ANStool and the BLUEnet tool support routers and switches. The BLUEnet tool also supports MPLS switches and routers. IaaS Framework supports routers and optical cards (switches). DRAC only supports switches, and AMPS only supports Cisco and Juniper routers. DRAGON supports GMPLS capable devices and non GMPLS switching devices.

Network connectivity at Layers 2, 2.5 and 3 is offered by ANStool and DRAGON. AutoBAHN and the BLUEnet tool offer connectivity at Layers 2 and 2.5, and IaaS Framework offers it at Layers 1, 2 and 3. The Juniper SRC offers connectivity at Layer 3, and DRAC at Layers 1 and 2.

ANStool and DRAGON therefore cover many of the Data Plane requirements, offering Ethernet, MPLS and IP, connectivity at Layers 2, 2.5 and 3, as well as support for routers and switches (MPLS in the case of DRAGON).

6.12 Layer 2 functionalities

Layer 2 requirements are focused on the configuration of VLANs, and the displaying of the list of these VLANs, VLAN trunks, Private VLANs, protocol based VLANs, configuration of GVRP for LAN interfaces, Q-in-Q and MAC-in-MAC.

Some tools (ARGON, BLUEnet and DRAC) can configure VLANs. BLUEnet can also display a list of them. AutoBAHN depends on the implementation of a Technology Proxy module and having support from network equipment.

BLUEnet and ARGON can work with VLAN trunks.

A combination of these tools would be required to fulfil all (or most of) the FEDERICA requirements.

6.13 Layer 2.5 functionalities

The basic requirements in Layer 2.5 are: configure MPLS, configure MPLS QoS, create L2 VPNs, create multipoint L2 VPNs and create L3 VPNs. ARGON and the BLUEnet tool can configure MPLS and MPLS QoS. The BLUEnet tool can also create L2 VPNs.

The only tool that fulfils all the FEDERICA requirements is DRAGON. Moreover, DRAGON is available from the project partners.

6.14 Layer 3 functionalities

The Layer 3 requirements are related to the version of IP (IPv4 or IPv6), the capability to request or configure specific topology, the support of current and new routing protocols, path discovery functionality and constraint based path computation.

AMPS supports IPv4. DRAGON has an IP Layer that provides for packet-based LSP capability. The Juniper SRC supports IPv4 for Juniper J/M/MX/T/TX and IPv4+IPv6 for Juniper E-series.

ARGON and PL-VINI are able to request or configure a specific topology. DRAGON allows end-to-end provisioning with a chosen topology. IaaS Framework (through MANTICORE) allows creating or deleting IP networks.

The current routing protocol in the BLUEnet tool is IS-IS; in DRAGON it is OSPF-TE. MANTICORE (IaaS Framework) uses OSPF, RIP and BGP, and PL-VINI supports whatever is supported by XORP (BGP, OSPF, etc.).

Path discovery is based on the Dijkstra algorithm for AMPS, and on spanning tree for Ethernet.

DRAGON incorporates TE constraints, AAA policy constraints, and time schedule constrains in path computation.

There is not a best tool to fulfil all FEDERICA Layer 3 functionalities. A combination of several tools would be needed.

6.15 Services Plane functionalities

The supported services by ARGON are sessions, pt-to-pt connections, and data transfer service. AutoBAHN provides a multi-domain BoD reservation service. DRAC offers services for creating paths with QoS parameters. DRAGON offers dynamic provisioning of end-to-end links with specific topology. IaaS Framework offers mechanisms for end users to create their own networks (with MANTICORE) or optical paths (Argia). Finally, the Juniper SRC offers time-of-day (automated), and (via a Web page) SOAP, Diameter or CORBA (manual) are services that can include CoS/QoS, policer, forward, drop and other policy actions.

7 Management of Virtualization Platforms

7.1 Introduction

The objective of this chapter is to explain the different management interfaces of XEN and VMware. These are the two potential Computing Element virtualization tools to be used within FEDERICA. For every tool and interface, its functionality and usage is documented and assessed. The main goal is to provide a quick view of every interface in order to facilitate the selection of one of the two tools, since they will need to be integrated with the toolbench to be finished during the second year of the project, and maybe also integrated with the tool or framework chosen from the previous study.

Deploying an infrastructure that is also composed of virtual machines adds a considerable amount of load, especially in case the administrator wants to manage it dynamically. The management of virtualized infrastructure includes several functionalities, such as:

- Discovery of the inventory of virtual computer systems
- Management of the lifecycle of virtual computer systems
- Creation/ modification /deletion of virtual resources
- Monitoring virtual systems for status and performance

7.2 Management of VMware

VMware provides a portfolio of different virtualization products suitable either for desktop usage (vmware player) or for enterprise virtualization services such as VMware Infrastructure 3 platforms (ESX Server, VirtualCenter Server systems and several additional server products for distributed resource management, disaster recovery, and high availability).

For VMware infrastructure components, the VMware Infrastructure object model is a comprehensive set of robust server-side composite objects that provides this “management layer” (referred to as “VMware Infrastructure Management”). This object model comprises data structures (composite object types) for managing, monitoring, configuring, obtaining information, and controlling life-cycle operations associated with virtual infrastructure. External clients can access this management framework through the VMware Infrastructure API.

The VMware Infrastructure management object model is instantiated on ESX Server systems and VirtualCenter Server systems. VirtualCenter Server has several additional capabilities beyond those of ESX Server (referred to as the “host agent”), most of which have to do with managing multiple host systems, or keeping historical data for multiple hosts: VirtualCenter has been designed to deploy, monitor, and manage multiple ESX Server host systems running any number of virtual machines. Of particular interest to FEDERICA is the capability of VirtualCenter for (i) deploying customized virtual machines from templates, and (ii) configuring and reporting on the status of various conditions within the datacenter.

VMware provides different flavours of APIs and SDKs. The Virtual Infrastructure (VI) management framework is accessed by external client applications using the VI API.

7.2.1 VI API

The VI API provides a complete set of language-neutral interfaces to the VMware Infrastructure Management framework. The VI API is implemented as industry-standard Web-Services, hosted on VirtualCenter Server and ESX Server systems. The VI API complies with the Web-Services Interoperability (WS-I) Organization Basic Profile 1.0, which includes XML Schema 1.0, SOAP 1.1 and WSDL 1.1.

The Web-Service provides all the operations necessary, including life-cycle operations, to monitor and manage virtual infrastructure components: compute resources, virtual machines, networks, storage, etc.

- Web-Services technology provides the necessary operations (using the same basic concept as “methods” in other programming languages). Using the VI SDK and the programming language of your choice, one can create client applications that invoke these operations to perform the full range of server-side management and monitoring tasks.
- The Web-Services API is defined in a WSDL (Web-Services Description Language) file. The WSDL file is used by Web-Services utilities to create client-side proxy code (stubs) that facilitate remote method invocation, and other low-level details of distributed object-oriented applications programming.
- Client applications invoke operations by sending SOAP (Simple Object Access Protocol)-formatted messages. One of the jobs of the client-side Web-Services tools is formatting (transparent to the developer) the SOAP messages from the programming language used.
- Communications between client and server occur over HTTP or HTTPS

Working with the VMware Infrastructure Management object model requires some understanding of the different data structures upon which the model depends. Central to the object model are the managed object types that provide system-wide services, and that support the creation and management of inventory objects. A managed object type is a server-side data structure that comprises properties and operations available on the server. Different managed objects offer different services (operations, methods). The various managed object types on the server define common administrative and management services one would expect to find in any datacenter - services, such as: managing performance (PerformanceManager), finding entities that exist in the inventory (SearchIndex), disseminating and controlling licences (LicenceManager), and configuring alarms to respond to certain events (AlarmManager).

7.2.1.1 Service Operations with the VI API

7.2.1.1.1 Monitoring and Performance management

VMware products provide a system of both built-in and customizable counters for capturing performance data. The VirtualCenter Server maintains historical performance data for the ESX Servers that it manages, in its database. In addition to using the default performance intervals, one can define one’s own. Clients can collect performance data on the host virtual machines, resource pools, or clusters (VirtualCenter only). This data includes CPU and memory utilization, network and disk performance data, floppy and CD-ROM drive performance, etc. It is possible to specify the frequency of updates (*perf intervals*), the number of samples in each update, and the performance data of interest to the client. In addition, one can retain a history of the performance data. Furthermore it is possible to create alarms, based on the measured data, so that the administrator can keep track of the resources being used.

7.2.1.1.2 Service Plane operation

Of particular importance to FEDERICA is the capability to manage virtual machines from the programming interface point of view, so that it is possible to create a machine from scratch or from a template. Given a virtual machine, the programmer can either power it on or off, suspend it, or change its resources ranging from available memory and CPU. It is also possible to define the network characteristics in terms of addressing and resource (peak, average bps and burst size).

7.2.2 VI Perl Toolkit

The VI Perl Toolkit lets the user automate a wide variety of administrative, provisioning, and monitoring tasks in the VMware Infrastructure environment. The client-side Perl framework provides an easy-to-use scripting interface to the VMware Infrastructure API. It can be used with ESX Server 3.x,VirtualCenter 2.x, and subsequent VMware Infrastructure 3 releases.

When one runs a VI Perl Toolkit script, the goal is always to access and potentially analyze or modify server side objects. One needs the name of the VI API objects and often their properties and method names. For example, if one wants to power down a virtual machine, one must know how to find the corresponding object, what the name of the power down method is, and how to call that method. The Managed Object Browser (MOB) allows the user to browse all objects on a remote host. The MOB lets one explore the objects on the system and obtain information about available properties and methods.

7.2.3 VIX API

The VIX API allows users to automate virtual machine operations on the VMware Server or VMware Workstation (VMware Server 1.1 and Workstation 6). To work with a virtual machine, one must first connect to the host where the virtual machine is stored. Typical operations using the VIX API are: connecting to a host, registering a virtual machine, starting or resuming a virtual machine and virtual machine guest operations. All functions that execute in a guest Operating System require VMware tools to be installed and running. All functions that modify a file system in the guest Operating System or that execute code in a guest Operating System require the client to authenticate as a user account known to the system.

7.2.4 VMware CIM APIs

CIM APIs are programmatic interfaces that enable client applications to use the industry-standard Common Information Model (CIM) for datacenter management. The VMware CIM APIs lets one view virtual machines and associated resources using profiles defined by the Storage Management Initiative Specification (SMI-S) of the Storage Networking Industry Association (SNIA), and to manage hosts using the System Management Architecture for Server Hardware (SMASH) of the Distributed Management Task Force (DMTF). ESX Server 3.x and ESX Server 3i support only Virtual Disk APIs and SDKs

7.2.5 VMware Guest SDK

A read-only API that enables management agents or other software running on a guest Operating System (Windows or Linux installed on a virtual machine running on an ESX Server host system) to collect various statistics, such as virtual machine memory usage, CPU speed and shares, and elapsed time since last virtual machine power-on or reset.

7.2.6 VMware VMCI SDK

The Virtual Machine Communication Interface (VMCI) is an experimental API for fast, efficient communication between a virtual machine and the host Operating System, and between two or more virtual machines on the same host. Bindings are available for C and C++, running on Workstation 6 (with VMware Tools installed).

7.2.7 VMware CIM SMASH

ESX Server 3i version 3.5 provides a CIM Object Manager (CIMOM) that implements a set of server discovery and monitoring features compatible with the SMASH standard. The VMware CIM SMASH API allows clients using industry-standard protocols to:

- Enumerate the system resources
- Monitor system status
- Power off host systems for maintenance

The VMware implementation of the SMASH standard leverages the open-source implementation of the Open Management with CIM (OMC) project. OMC is a collaborative effort bringing together developers from a variety of organizations to provide tools and software infrastructure for hardware vendors and others who need a reliable implementation of the Distributed Management Task Force (DMTF) management profiles. The VMware CIM SMASH API supports the following protocols:

- CIM-XML over HTTP (WBEM)
- WS-MAN
- SLP (SMASH)

The VMware CIM SMASH API supports a number of profiles defined by the SMWG. These profiles have overlapping structures, and can be used individually (sometimes) or in combinations to manage a server. The basic functionality for configuring the state of a remote server is provided through profiles. Of particular importance to FEDERICA is the Ethernet Port, IP interface and System Memory Profile which let one manage these resources.

7.2.8 Related Standards

The Distributed Management Task Force (DMTF) is an industry organization made up of member companies to develop and promote a standard method for systems management. The method that the DMTF derived is called the CIM (Common Information Model). CIM is an object oriented model to represent a wide variety of systems in a standard and neutral way, and is commonly referred to as the CIM schema.

According to the CIM schema, a common component such as a server, or a network router is represented in a way that all management tools that use CIM will understand. The CIM standard has been used by all the major systems management tools available today. The CIM standard also has a way to represent management data, but there are many different ways that the data can be accessed.

The Distributed Management Task Force (DMTF) with the System Virtualization, Partitioning and Clustering Working Group has developed a set of documents which help to focus the standardization effort in the Virtualization industry. In the DMTF, the CIM schema and an associated behaviour for a particular management domain is defined through a series of management profile documents. Each profile identifies the classes, properties, methods, and values that should be instantiated and manipulated to represent and manage a given domain.

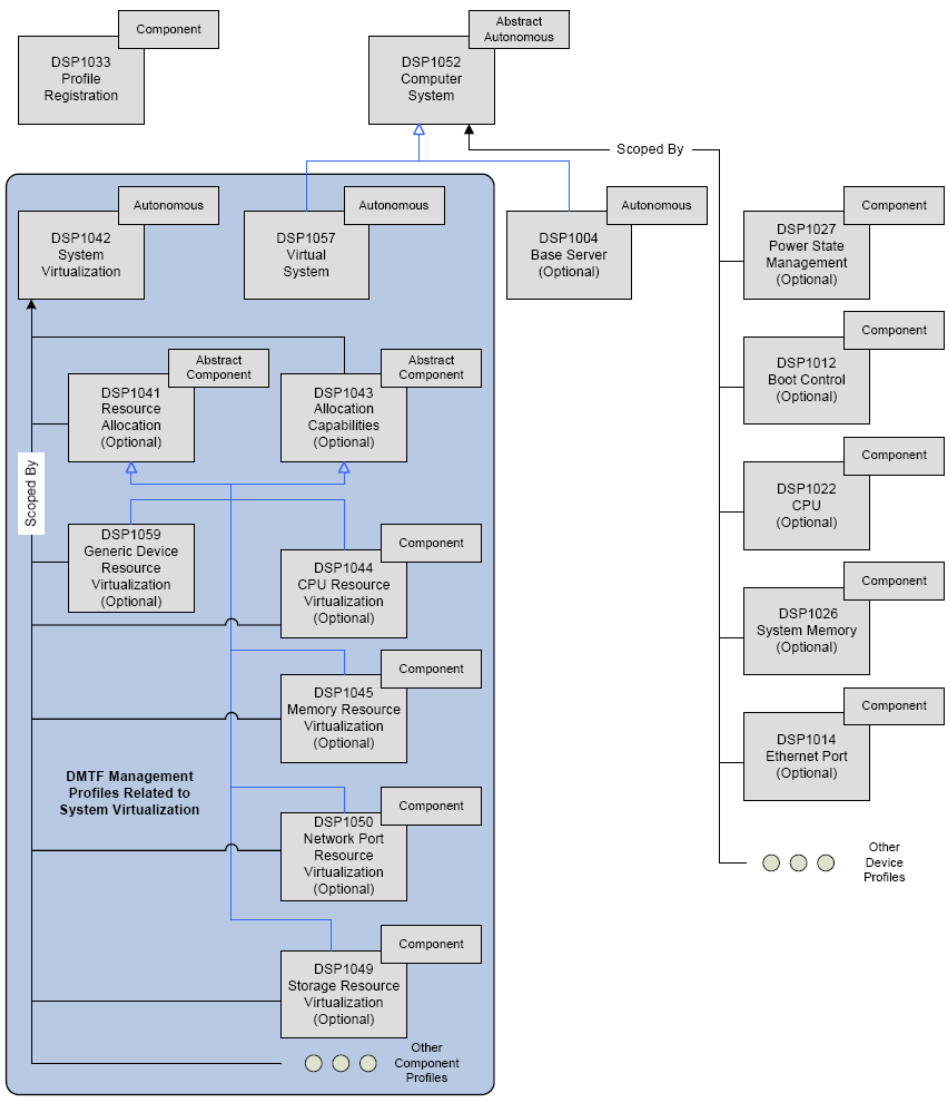

Figure 5: Structure of the profile documents related to virtualization

The previous figure (Figure 5) shows the structure of the profile documents related to virtualization. Two abstract profiles, Resource Allocation Profile and Allocation Capabilities Profile, describe the basic abstract patterns used for the management of virtual systems. Two top-level, autonomous profiles define the Computer System Profile: System Virtualization Profile (SVP) and Virtual System Profile. A series of device-specific profiles describe in more detail the management of virtual devices.

The Resource Allocation Profile[1] describes the basic resource allocation pattern for resource pools, allocations, and setting data. It also defines the resource-pool-lifecycle management and relationships.

The Allocation Capabilities Profile[2] extends the management capability of the referencing profiles, by adding the ability to represent property values for resource allocation requests for a given resource.

The System Virtualization Profile[3] is an autonomous profile that specifies the object model needed for the representation of host systems and the discovery of hosted virtual computer systems. In addition, it specifies a service for the manipulation of virtual computer systems and their resources, including operations for the creation, deletion, and modification of virtual computer systems and operations for the addition or removal of virtual resources to or from virtual computer systems.

The Virtual System Profile[4] is an autonomous DMTF management profile that defines the model needed to provide for the inspection of a virtual system and its components. The Virtual System Profile specifies the Computer System Profile that defines the model needed to define a basic computing platform. In addition, the Virtual System Profile defines optional basic control operations for activating, deactivating, pausing, or suspending a virtual system.

The Computer System Profile references a set of component profiles that are defined for each of the device types that make up a computer system including CPU, memory, storage (block and file backed) and storage adapters, networking and networking adapters, removable devices, keyboard, video and mouse devices.

7.2.8.1 Web Based Enterprise Management

Web Based Enterprise Management (WBEM) is a standard of the DMTF. The first major component of a WBEM implementation is the Common Information Model Object Manager (CIMOM). CIMOM is an implementation concept and is the core engine that holds the CIM data. On top of the CIMOM is the WBEM interface that is basically a HTTP server, but not one used for Web browsing, as the WBEM interface has its own unique port number 5988. Therefore, in order to communicate with the CIMOM, one has to use a CIM WBEM client that can communicate over the standard port. Generally, the CIM WBEM client represents the system management system console, and can be used to configure all WBEM-enabled systems. A CIM Client issues CIM Operation requests and receives and processes CIM Operation responses. A CIM Server node receives and processes CIM Operation requests, coordinates the processing of requests and responses among the CIM Providers and sends CIM Operation responses back to the CIM Client. A CIM Provider translates CIM-formatted requests into resource specific operations and translates resource-specific responses into CIM-formatted responses.

CIM-XML is currently the only standard-based protocol for exchanging CIM information. CIM-XML uses xmlCIM as the payload and HTTP as the transport. CIM-XML defines all interactions between CIM products as CIM messages.

43

[1] http://www.dmtf.org/standards/published_documents/DSP1041_1.1.0.pdf
[2] http://www.dmtf.org/standards/published_documents/DSP1043.pdf
[3] http://www.dmtf.org/standards/published_documents/DSP1042.pdf
[4] http://www.dmtf.org/standards/published_documents/DSP1057.pdf

7.2.8.2 SMASH

The DMTF's Systems Management Architecture for Server Hardware (SMASH) initiative has produced a suite of specifications that deliver architectural semantics, industry standard protocols and profiles to unify the management of the data center. The SMASH Server Management (SM) Command Line Protocol (CLP) specification enables simple and intuitive management of heterogeneous servers in the data center. The CLP is defined as messages going over the wire, not a command line interface. CLP is a command response protocol which is case insensitive, with reserved characters (like " , -, and /) and reserved keywords The SMASH initiative takes full advantage of the DMTF's Web-Services for Management (WS-Management) specification - delivering standardised Web-Services management for server environments. Both provide server management independent of machine state, Operating System state, server system topology or access method, facilitating local and remote management of server hardware. The SMASH initiative also includes the SM Managed Element Addressing Specification, SM CLP-to-CIM Mapping Specification, SM CLP Discovery Specification, and SM Profiles.

7.2.8.3 Web-Services for Management (WS-Management) Specification

The Web-Services for Management (WS-Management) specification describes a general Web-Services protocol based on SOAP for managing systems such as PCs, servers, devices, Web-Services and other applications, and other manageable entities. Services can expose only a WS-Management interface or compose the WS-Management service interface with some of the many other Web-Service specifications.

A crucial application for these services is in the area of systems management. To promote interoperability between management applications and managed resources, this specification identifies a core set of Web-Service specifications and usage requirements that expose a common set of operations central to all systems management.

7.3 Management of the XEN tool

There are three different systems for the management of the XEN tool: **Xend**, **xm** and **management API**.

7.3.1 Xend

The Xend node control daemon performs system management functions related to virtual machines. It forms a central point of control of virtualized resources, and must be running in order to start and manage virtual machines. Xend must be run as root because it needs access to privileged system management functions.

Xend can also be started from the command line, and supports the following set of parameters:

xend start start xend, if not already running

xend stop stop xend if already running

xend restart restart xend if running, otherwise start it

xend status indicates xend status by its return code

An HTTP interface and a Unix domain socket API are available to communicate with Xend. This allows remote users to pass commands to the daemon. By default, Xend does not start an HTTP server. It does start a Unix domain socket management server, as the low level utility xm requires it. For the support of cross-machine migration, Xend can start a relocation server. This support is not enabled by default for security reasons.

Note: the example Xend configuration file modifies the defaults and starts up Xend as an HTTP server as well as a relocation server.
From the file:

#(xend-http-server no)

(xend-http-server yes)

#(xend-unix-server yes)

#(xend-relocation-server no)

(xend-relocation-server yes)

Comment or uncomment lines in that file to disable or enable features that you require.

Connections from remote hosts are disabled by default:

Address xend should listen on for HTTP connections, if xend-http-server # set.

Specifying 'localhost' prevents remote connections.

Specifying the empty string '' (the default) allows all connections.

#(xend-address '')

(xend-address localhost)

It is recommended that if migration support is not needed, the *xend-relocation-server* parameter value be changed to .no. or commented out.

The operational status of the above interface cannot be guaranteed. It is old and generally working on an SXP-based interface, on port 8000.

There are also two more Xend interfaces, but without documentation:

- *xend-unix-xmlrpc-server*: Legacy XML-RPC server, over HTTP/unix, the recommended way to access Xend in 3.0.4.
- *xend-tcp-xmlrpc-server*: Ditto, over TCP, on port 8006.

7.3.2 Xm

The xm tool is the primary tool for managing XEN from the console. The general format of an xm command line is:

xm command [switches] [arguments] [variables]

The available *switches* and *arguments* are dependent on the *command* chosen. The *variables* may be set using declarations of the form variable=value and command line declarations override any of the values in the configuration file being used, including the standard variables described above and any custom variables (for instance, the xmdefconfig file uses a vmid variable).

7.3.3 XEN-Management-API

The XEN Management API is an interface for remotely configuring and controlling virtualised guests running on a XEN-enabled host.

The API is presented here as a set of Remote Procedure Calls, with a wire format based upon XML-RPC. No specific language bindings are prescribed.

The API reference uses the terminology classes and objects. For our purposes a class is simply a hierarchical namespace; an object is an instance of a class with its fields set to specific values. Objects are persistent and exist on the server-side.

Clients may obtain opaque references to these server-side objects and then access their fields viaget/set RPCs.

For each class we specify a list of fields along with their types and qualifiers. A qualifier is one of:

- ROrun: the field is Read Only. Furthermore, its value is automatically computed at runtime. For example: current CPU load and disk IO throughput.
- ROins : the field must be manually set when a new object is created, but is then Read Only for the duration of the object's life. For example, the maximum memory addressable by a guest is set before the guest boots.
- RW: the field is Read/Write. For example, the name of a VM.

The following classes are defined:

Name	Description
session	A session
task	A long-running asynchronous task
event	Asynchronous event registration and handling
VM	A virtual machine (or 'guest')
VM metrics	The metrics associated with a VM
VM guest metrics	The metrics reported by the guest (as opposed to inferred from outside)
host	A physical host
host metrics	The metrics associated with a host
host cpu	A physical CPU
network	A virtual network
VIF	A virtual network interface
VIF metrics	The metrics associated with a virtual network device
PIF	A physical network interface (note separate VLANs are represented as several PIFs)
PIF metrics	The metrics associated with a physical network interface
SR	A storage repository
VDI	A virtual disk image
VBD	A virtual block device
VBD metrics	The metrics associated with a virtual block device

PBD	The physical block devices through which hosts access SRs
crashdump	A VM crashdump
VTPM	A virtual TPM device
console	A console
user	A user of the system
debug	A basic class for testing

With the following relations:

object.field	object.field	relationship
host.PBDs	PBD.host	many-to-one
SR.PBDs	PBD.SR	many-to-one
VDI.VBDs	VBD.VDI	many-to-one
VDI.crash_dumps	crashdump.VDI	many-to-one
VBD:VM	VM.VBDs	one-to-many
crashdump.VM	VM.crash_dumps	one-to-many
VIF.VM	VM.VIFs	one-to-many
VIF.network	network.VIFs	one-to-many
PIF.host	host.PIFs	one-to-many
PIF.network	network.PIFs	one-to-many
SR.VDIs	VDI.SR	many-to-one
VTPM.VM	VM.VTPMs	one-to-many
console.VM	VM.consoles	one-to-many
host.resident_VMs	VM.resident_on	many-to-one
host.host_CPUs	host_cpu.host	many-to-one

An example of the different fields for the class VM is shown below:

Quals	Fields	Type	Description
ROrun	uuid	string	unique identifier/object reference
ROrun	power_state	vm_power_state	Current power state of the machine
ROrun	allowed_operations*	(vm_operations) Set	list of the operations allowed in this state. This list is advisory only and the server state may have changed by the time this field is read by a client
ROrun	current_operations*	(string -> vm_operations) Map	links each of the running task using this object (by reference) to a current_operation enum which describes the nature of the task
RW	name/label	string	a human-readable name
RW	name/description	string	a notes field containing human-readable description
RW	user_version	int	a user version number for this machine

RW	is_a_template	bool	true if this is a template. Template VMs can never be started, they are used only for cloning other VMs
RW	auto_power_on**	bool	tru if this VM should be started automatically after host boot
ROrun	suspend_VDI	VDI ref	The VDI that a suspend image is stored on. (Only has meaning if VM is currently suspended)
ROrun	resident_on	host ref	the host the VM is currently resident on
ROrun	scheduled_to_be_resident_on*	host ref	the host on which the VM is due to be started/resumed/migrated. This acts as a memory reservation indicator
RW	affinity*	host ref	a host which the VM has some affinity for (or NULL). This is used as a hint to the start call when it decides where to run the VM. Implementations are free to ignore this field
RW	memory/static_max	int	Statically-set (i.e. absolute) maximum (bytes)
RW	memory/dynamic_MAX	INT	Dynamic maximum (bytes)
RW	memory/dynamic_min	int	Dynamic minimum (bytes)
RW	memory/static_min	int	tacitly-set (i.e. absolute) minimum (bytes)
RW	VCPUs/params	(string->string)Map	configuration parameters for the selected VCPU policy
RW	VCPUs/max	int	Max number of VCPUs
RW	VCPUs/at_startup	int	Boot number of VCPUs
RW	actions/after_shutdown	on_normal_exit	action to take after the guest has shutdown itself
RW	actions/after_reboot	on_normal_exit	action to take after the guest has rebooted itself
RW	actions/after_crash	on_crash_behaviour	action to take if the guest crashes
ROrun	consoles	(console ref) Set	virtual console devices
ROrun	VIFs	(VIF ref) Set	virtual network interfaces
ROrun	VBDs	(VBD ref) Set	virtual block devices
ROrun	crash_dumps	(crashdump ref) Set	crash dumps associated with this VM
ROrun	VTPMs	(VTPM ref) Set	crash dumps associated with this VM
ROrun	VTPMs	(VTPM ref) Set	virtual TPMs
RW	PV/bootloader	string	name of or path to bootloader
RW	PV/kernel	string	path to the kernel
RW	PV/kernel	string	path to the kernel
RW	PV/ramdisk	string	path to the initrd
RW	PV/args	string	kernel command-line arguments

RW	PV/bootloader_args	string	miscellaneous arguments for the bootloader
RW	PV/legacy_args*	string	to make Zurich guests boot
RW	HVM/boot_policy	string	HVM boot policy
RW	HVM/boot_params	(string->string) Map	HVM boot parameters
RW	HVM/shadow_multiplier*	float	multiplier applied to the amount of shadow that will be made available to the guest
RW	platform	(string->string) Map	platform-specific configuration
RW	PCI-bus	string	PCI bus path for pass-through devices
RW	other_config	(string->string) Map	additional configuration
ROrun	domid	int	domain ID (if available, 01 otherwise)
ROrun	domarch*	string	Domain architecture (if available, null string otherwise)
ROrun	last_boot_CPU_flags*	(string->string) Map	describes the CPU flags on which the VM was last booted
ROrun	is_control_domain	bool	true if this is a control domain (domain 0 or a driver domain)
ROrun	metrics	VM_metrics ref	metrics associated with this VM
ROrun	guest_metrics	VM_guest_metrics ref	metrics associated with the running guest
ROrun	last_booted_record*	string	marshalled value containing VM record at time of last boot, updated dynamically to reflect the runtime state of the domain
RW	recommendations*	string	An XML specification of recommended values and ranges for properties of this VM
RW	xenstore_data*	(string->string) Map	data to be inserted into the xenstore tree (/local/domain/!domid¿/vm-data) after the VM is created

*Only enterprise edition
**Only community edition

For the creation and start of a virtual machine the next two RPC operations have to be called:

RPC name: *create*

Overview: Create a new VM instance, and return its handle.

Signature:

(VM ref) create (session_id s, VM record args)

Arguments:

Type	name	description
VM record	args	All constructor arguments

Return Type: VM ref

reference to the newly created object

RPC name: *start*

Overview: Start the specified VM. This function can only be called with the VM is in the Halted State.

Signature:

void start (session_id s, VM ref vm, bool start_paused, bool force)

Arguments:

Type	name	description
VM ref	vm	The VM to start
Bool	start_paused	Instantiate VM in paused state if set to true
Bool	force	Attempt to force the VM to start. If this flag is false then the VM may fail pre-boot safety checks (e.g. if the CPU the VM last booted on looks substantially different to the current one)

Return Type: void

Possible Error Codes: VM Bad Power State, VM HVM Required, VM is Template, Other Operation in Progress, Operation not Allowed, Bootloader Failed, Unknown Bootloader, No Hosts Available, Licence Restriction

Finally a VM has the following states in the API calls:

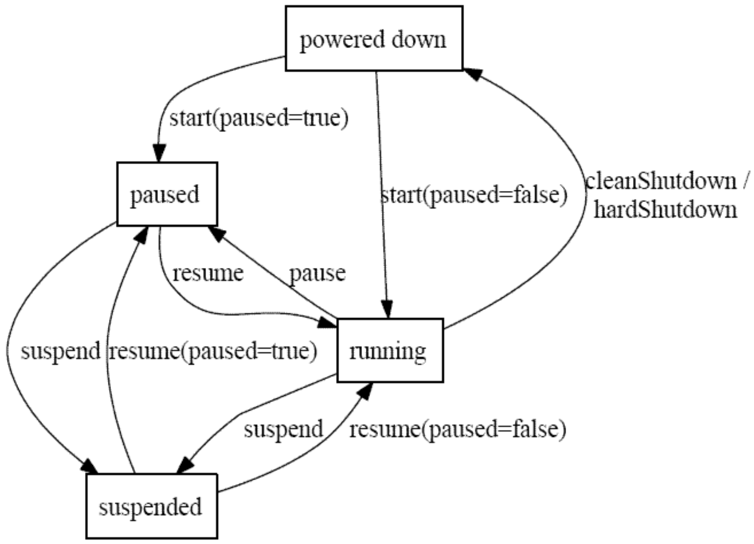

Figure 6: States in the API calls

7.3.4 Conclusions

Summarising XEN, we have:

- xm: is the primary tool for managing XEN from the console.
- xend-http-server: is a very old and non-guaranteed HTML interface, generally working on an SXP-based interface, on port 8000.
- xend-unix-server: Ditto, but using a Unix domain socket.
- xend-unix-xmlrpc-server: Legacy XML-RPC server, over HTTP/Unix, the recommended way to access Xend in 3.0.4.
- xend-tcp-xmlrpc-server: Ditto, over TCP, on port 8006.
- xen-management-api: All new XEN-API interface, available in preview form now, and expected to be part of release 3.0.5.

Therefore, for FEDERICA, the three important interfaces are:

- xend-unix-xmlrpc-server: this is the recommended way to access Xend.
- xend-tcp-xmlrpc-server: another way to access Xend by the tcp port.
- xen-management-api: a new API for remote management of the XEN tool based on XML-RPC.

At that moment not too much information is available for the xend xmlrpc interfaces. For the management API two documents are defined: one for the enterprise edition and another for the community edition. No examples are given for any of these interfaces - only some bindings for the management API.

Also, at that moment, none of these interfaces is using the standard CIM. It is supposed to be included in the next versions of the XEN management API.

7.3.5 Using MLN to manage XEN based Virtual Networks

Although not a straightforward management API for managing Virtual Hosts and networks, MLN (Manage Large Networks) is an option for managing large number of Virtual Hosts. MLN is a virtual machine administration tool designed to build and run virtual machine networks based on XEN and User-Mode Linux. MLN is CLI-based tool and a *configuration description language*.

As a virtual machine management configuration front end, MLN builds on top of concepts of Virtual Hosts, templates, virtual switches and Virtual Networks. These concepts are briefly described below, from the XEN-based configuration point of view.

7.3.5.1 Concepts

- A **Virtual Host:** consists of its own filesystem and runs in software on top of a specified OS. MLN will customize the filesystem for the Virtual Host.
- A **filesystem template:** is a basic pre-customized filesystem for a Virtual Host. Templates are available for download from the MLN homepage. Templates differ according to the distribution they are based upon and how much software they contain. For example, one template might be a user desktop environment with graphical login and numerous productivity applications, while another might be a small firewall environment using busybox to replace applications and conserve space. A virtual machine based on a template of this kind is the basis for what are called *virtual appliances* which can be specialized to perform specific tasks (as the examples later will show).
- MLN supports the **virtual switch** capability. For XEN, the switch is the so-called “Ethernet bridge-device” that can connect several network interfaces together like a switch.
- **Virtual Networks** are composed of virtual switches and Virtual Hosts which connect on them. Many Virtual Hosts and switches make large networks and it is MLN's job to configure and build these networks based on a configuration specification (i.e. language). It is possible to build automatically functional networks or it is possible to build lots of virtual machines that are connected to switches, and configure networking on them manually.
- Virtual Networks can also be part of real networks. This means that neither the physical hosts nor the Virtual Hosts can tell the difference, they are simply on the same LAN.
- One Virtual Network is called one “*project*”. MLN can build and run several *projects* at the same time. Sometimes it is sensible to keep them apart, other times one might wish to connect them together. *Projects* are identified by their name.

7.3.5.2 MLN configuration language

The MLN language looks much like a declarative programming language. The structure of the language is a hierarchical sequence of blocks containing keyword/value pairs. A block is enclosed in curly brackets ({ }). A keyword is generally expressed in the form **keyword value** but is not bound to it. Some keywords are lines with several parameters. It is often natural to place one keyword/value per line, but semicolons can be used to place several pairs on the same line.

Each host and network switch will have one block. All hosts in a *project* do not have to be connected in the same network.

A *project* has no restrictions regarding the number of hosts or switches. The only mandatory part of a *project* description is a block of global definitions with at least the name of the *project*. *Project* names must be unique for each user. Definitions of one or more hosts and perhaps network switches constitute the network topology. Hosts can have several network interfaces that can be assigned to switches in arbitrary topologies.

7.3.5.3 Examples

Here is an example of a ring-topology:

```
global {
       project ring
}
host router1 {
    network eth0 {
        switch A
   }
   network eth1 {
        switch B
   }
}
host router2 {
    network eth0 {
         switch B
    }
    network eth1 {
         switch C
    }
}
host router3 {
    network eth0 {
         switch C
    }
    network eth1 {
         switch A
    }
}
switch A { }
switch B { }
switch C { }
```

or another simple example

```
global {
      project example1
}
host fish {
      network eth0 {
                  switch lan
                  address 10.0.0.1
                  netmask 255.255.255.0
                  }
      }

host chips {
            network eth0 {
            switch lan
            address 10.0.0.2
            netmask 255.255.255.0
            }
```

```
      }

switch lan {}
```

For larger projects it is possible to define superclasses and variables

```
global {
            project example2
}
superclass common {
            xen
            template ubuntu-server.ext3
            term screen
            memory 64M
            nameserver 128.39.89.10
            network eth0 {
                  switch lan
                  netmask 255.255.255.0
            }
}

host gateway {
      superclass common
      network eth1 {
      address dhcp
      }

      network eth0 {
      address 10.0.0.1
      }
      startup {
      iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
      echo 1 > /proc/sys/net/ipv4/ip_forward
      }
}

host backend {
      superclass common
      network eth0 {
                        address 10.0.0.2
                        gateway 10.0.0.1
      }
}

switch lan { }
```

MLN also provides a network daemon for distribution of projects among several physical servers. A physical server is in MLN called a **service_host** because it provides a hosting service to the virtual machine. A physical host must be made aware of it being a service host in its local MLN configuration files.

A virtual machine is assigned a service host in the following way:

```
host startfish {
     service_host server1.it.FEDERICA.eu
     memory 96M
     network eth0 {
             address dhcp
     }
}
```

7.3.5.4 Miscellaneous

MLN does not intent to re-invent configuration management paradigms in its own language. The plug-in architecture extends other configuration management tools to be integrated as easily as possible with MLN. A plug-in that is executed can do two actions: access the entire MLN data tree and change the project before it is built, or configure virtual machine filesystems during the build process.

A typical example:

```
suberclass common {
    template ubuntu-server-cfengine.ext3
    cfagent {
       control:
         any::
          actionsequence =
            (
            shellcommands
            processes
            )
    }
}
host agent {
    cfagent {
      shellcommands:
          "/usr/bin/updatedb"
      processes:
         "cron" signal=hup
    }
}
```

A plug-in in the Perl programming language that writes the above specified **cfagent** into a file does not have to be more than the following code:

```
sub cfenginePlugin_configure {
  my $hostname = $_[0];
  if ( getScalar("/host/$hostname/cfagent")){
    my @cfagent_poligy =
          getArray("/host/$hostname/cfagent");
    writeToFile($hostname,
              "/cfengine/inputs/cfagent.conf",
              @cfagent_policy);
  }
}
1;
```

The added value of this approach is that it is relatively easier to combine MLN with tools that the community is experienced with and that can handle long-term management of hosts, instead of developing new ones from scratch.

7.4 Using Libvirt to manage Virtual Hosts

Libvirt is a **CIM provider** for managing linux virtualization platforms. The libvirt-cim provider depends on the DMTF CIM v2.16 schema[5]. This schema incorporates a number of enhancements and additions to support alliance partners and DMTF Profiles for new Management Initiatives (i.e. Virtualization, PCI Devices, KVM Redirections, LED Modelling, Sensor Updates and security modelling. It is written in C and should work with any CIMOM that supports CMPI 2.0 providers. The intent is to implement the SVP virtualization class model currently available. Libvirt supports the:

- XEN hypervisor (on Linux and Solaris),
- QEMU emulator
- KVM Linux hypervisor
- LXC Linux container system
- OpenVZ Linux container system

Furthermore Libvirt supports storage on IDE/SCSI/USB disks, FibreChannel, LVM, iSCSI, NFS and filesystems. It, also, provides:

- Remote management using TLS encryption and x509 certificates
- Remote management authenticating with Kerberos and SASL
- Local access control using PolicyKit
- Management of virtual machines, Virtual Networks and storage

Although the premise for Libvirt seems simple, the project has turned out some very mature features and tools, including:

- Local administration tool, including a shell (virsh), a GNOME application, and a GNOME monitoring applet.
- Many control interfaces - shell scripting, Python and Perl bindings, and robust APIs.
- Monitoring interfaces - feeding stats and states to applications, daemons, and the API hooks for other applications to utilize
- A robust policy framework - enabling complex policies to monitor, control, and correct domains running on the node.
- An XML structure for defining domains, which is portable, easily parsed, and human readable.

56

[5] http://www.dmtf.org/standards/cim/cim_schema_v216/

8 Conclusions

This deliverable has proposed a set of new basic definitions and virtualization building blocks to use when dealing with Virtual Network architectures. This taxonomy will be used as a common nomenclature for the whole FEDERICA project, and covers all of its infrastructure available resource

The work done during the first period of the project has been focused on designing the building blocks needed to define a virtual architecture for the creation of virtual slices. A set of FEDERICA requirements for virtualization management purposes has been defined in this deliverable, according to a number of potential Use Cases. These Use Cases represent both the user/researcher and FEDERICA NOC Infrastructure points of view, and have been internally identified in order to map FEDERICA capabilities with user needs. A more extended definition of Use Cases - focused on real scenarios to develop by FEDERICA - is being prepared in the SA project activity.

As currently there is no effort in standardization bodies on network virtualization, the FEDERICA consortium will establish contacts with communities interested on this topic in order to move forward a standardization process for the virtual building blocks definitions, and the associated architectures. Discussions will be started e.g. with people from the Network Markup Language (NML) working group of the OGF. This discussion can help on the modelling process for the dynamic creation of Virtual Network slices for FEDERICA infrastructure users. The NML group is mainly focused on multi-layer network modelling and the ability to describe an abstracted view of the network, with capability information, pointers to services for policy decision and policy enforcement points.

With the set of basic definitions, and a virtual architecture in place, we then made a in-depth analysis of existing tools and frameworks, which have some functionalities that could be used within the FEDERICA project. These tools are candidates for the toolbench development and for the process of creating virtual slices, since FEDERICA aims at identifing available tools or frameworks to be used as starting point.

Regarding the analysed tools, we can conclude that probably a combination of them will be needed to fulfil all the requirements. This is because there is no unique tool able to do everything by itself; each one covers a different set of FEDERICA requirements. However, a few of them have been highlighted as the most suitable. These are: MANTICORE with the IaaS Framework, BLUENet, DRAGON and PL-VINI. During the next period more effort will be dedicated to the features that these tools can provide. The complete study of the FEDERICA requirements and the capabilities of the candidate tools is documented in Annex I of this deliverable. A comparison study of the tools can also be found in this Annex I

In order to integrate Computing Elements into the FEDERICA virtual slices, this deliverable also presents a study of different management interfaces that XEN and VMware have (two of the potential tools for virtualization to be used within FEDERICA) . This is because it is interesting to use the tool that has the most scalable API for its integration with the FEDERICA toolbench. As a best initial choice we will use VMware.

The study on the possibilities of each tool and the best way to cover all the FEDERICA requirements will continue to be studied in the Joint Research Activity JRA1 and tested in the Service Activity SA2, as an ongoing effort. As a general conclusion, the next step in the research is towards pushing the standardization efforts of the definition provided for the building blocks and its architecture, while starting to work on identifying the way to integrate the selected tools within FEDERICA.

9 References

Analyzed Tools/Frameworks

AMPS:

http://www.geant2.net/server/show/nav.1798

http://www.terena.org/events/tnc2006/programme/presentations/show.php?pres_id=244

ANStool:

http://anstool.grnet.gr:6080/

ARGON:

http://www.viola-testbed.de/
fileadmin/VIOLA/ws06/Dynamic_Allocation_of_Network_Resources_20060322_Moll_.pdf

http://web.informatik.uni-bonn.de/IV/martini/Forschung/Projekte/index-en.html

http://www.viola-testbed.de/fileadmin/VIOLA/reports/B2-4-1-allocation.pdf

AutoBAHN:

http://www.geant2.net/server/show/nav.756

BLUEnet:

http://www.terena.org/activities/tf-ngn/tf-ngn21/reijs-bluenet.pdf

DRAC:

http://www.nortel.com/solutions/optical/collateral/nn110181.pdf

http://www.nortel.com/corporate/news/collateral/ntj1_project_drac.pdf

DRAGON:

http://dragon.east.isi.edu/

http://dragon.maxgigapop.net/twiki/bin/view/DRAGON/WebHome

G3:

http://www.cesnet.cz/doc/techzpravy/2005/g3/

http://www.ces.net/netreport/hep-cesnet-experimental-facility/

GINS:

http://www.garr.it/

https://www.noc.garr.it/GINS/home_statistics.php

IaaS Framework:

http://www.iaasframework.com/

Juniper SRC:

http://www.juniper.net/products_and_services/session_and_resource_control/

PL-VINI:

http://www.vini-veritas.net/

Management of Virtualization Platforms

White Papers:

http://www.dmtf.org/standards/published_documents/DSP2013_1.0.0.pdf
http://www.vmware.com/support/developer/vc-sdk/visdk25pubs/visdkprogrammingguide.pdf

XEN Enterprise Management Interface:
http://community.citrix.com/download/attachments/17596457/xenenterpriseapi4_1.pdf?version=1

XEN Management Interface:
http://wiki.xensource.com/xenwiki/XenApi?action=AttachFile&do=get&target=xenapi-1.0.0.pdf

XEN Summmit API Slides:
http://wiki.xensource.com/xenwiki/XenApi?action=AttachFile&do=get&target=Xen+Summit+API+Slides+2007-04-18.pdf

XEN Manual:
http://bits.xensource.com/Xen/docs/user.pdf

MLN:

The MLN manual http://mln.sourceforge.net/doc/mln-manual.html
http://www.usenix.org/event/lisa06/tech/full_papers/begnum/begnum_html/index.html

Libvirt:
http://libvirt.org/

ANNEX I

1 Tools and Frameworks comparison

This Annex documents the study of the different tool and frameworks selected as candidates tools for FEDERICA. The study is divided into different topics, which are analysed for each one of the candidate selected tools and framework. The topics considered for the analysis are: Control Plane functionalities and Management Plane functionalities

1.1 Control Plane functionalities of the Tools/Frameworks

In this section all the tools/frameworks introduced in the main part of the deliverable are compared from the point of view of the Control and Signalling Plane functionalities. The functionalities that the Control Plane should have are also introduced.

Global performance includes characteristics such as, if the Control Plane is intra- or inter-domain, centralized or distributed, or if the provisioning is automated or manual. Another important aspect is the topology. Regarding the topology it is of great interest how it is modelled in the Control Plane, how it is provided, or how the discovery is done, what kind of information about the intra-domain topology is shared between different domains, etc. Routing and path discovery and computations are other issues that have been taken into account, as well as the signalling functionalities; the type of inter- and intra-domain signalling implemented; the connection, protection and restoration signalling and the QoS provisioning.

1.1.1 AMPS

AMPS provides the Control Plane for Inter-domain and Intra-domain network environment. The two services that carry out the necessary tasks are the Inter-domain Module and the Intra-domain Module: The first service exchanges messages with AMPS servers in neighbouring domains, receiving and forwarding PIP requests. The second service stores information about accepted PIP reservations, and decides whether new requests can be satisfied or not.

The AMPS system follows a distributed model of control. Network equipment is configured separately in each domain, and the provisioning mode can be manual or automatic depending on the domain policy.

AMPS has a dedicated service called Network Information Service (NIS) that serves as a repository of network information. It is responsible of calculating paths for a particular flow within the local domain. To do this, the database contains a record of each link and its associated metrics. In order to properly calculate the path, the Intra-domain Module has to supply the start-point and the end-point of the flow to the NIS. In return, NIS provides the Intra-domain Module with the collection of links composing a path for the flow. The NIS currently only works for networks using link-state IGPs.

AMPS also has the NIS-NDM (Network Discovery Module). This module allows the AMPS administrator to semi-automatically discover their network's IP topology using SNMP, SSH or telnet. If the AMPS server is not able to use SSH or telnet for some reason, the Web interface allows the operator to manually add router interface IGP routing values.

AMPS does not share topology information, it is just propagated by the routing network equipment. The only information shared is the location of the AMPS manager of peering domains. The system also provides a dedicated Pathfinder module for each domain manager to compute its own path

(Dijkstra algorithm). It constructs the inter-domain path by computing parts from each domain along the path. Each domain knows only neighbour domains and contacts them during the path computation and establishment. The path signalling is based on a RSVP-like approach. AMPS does not provide intra-domain signalling, however it supplies the signalling for connection provisioning. The QoS provisioning is given by Premium IP.

1.1.2 ANStool

ANStool works mainly in the OAM Plane. It does not communicate directly with the Control Plane of network devices.

1.1.3 ARGON

As a centralized entity, ARGON controls the network resources to which it has been assigned by directly configuring the network devices. However, ARGON can be integrated into a multi-domain environment by means of its Harmony/NSP interface developed in the EC-IST project Phosphorus. Using ARGON, it would be possible to operate several domains, each controlled by an ARGON instance (or by another system providing the same inter-domain interface, i.e. Argia or DRAC).

1.1.4 AutoBAHN

AutoBAHN provides end-to-end bandwidth on demand reservation services implemented over a set of administrative domains which collaborate on a peer-to-peer basis. Its control is distributed, and therefore a domain manager is needed in each domain.

Each Manager is split into two modules. The Inter-Domain Manager module is responsible for the inter-domain circuit reservations on behalf of a domain. This includes inter-domain communication, resource negotiations with adjacent domains, resource scheduling, and topology advertisements. The second module is the Domain Manager (DM) module which is responsible for the instantiation of BoD instances within a single domain. The DM has a detailed knowledge of the topology of its domain. It participates in the inter-domain path finding process by examining the feasibility of providing an end-to-end path within its local domain. It contacts the Technology Proxy module to request the configuration of the BoD service instance. A Technology Proxy module, allows the AutoBAHN system to support a wide range of technologies and vendors according to multiple domain and global requirements.

The provisioning is done automatically by the system and each reservation has start and end time attributes attached. The resources are booked in the calendar module, not configured.

Two domain topologies are used by the system. The intra-domain is a Technology Specific Topology model which contains technical details and shows the physical connections between switches. This domain is available only for the DM module. Currently Ethernet and SDH topologies are implemented. The Specific Topology is provided manually; the database has to be filled with details regarding the physical topology. The users of AutoBAHN can define this network details using a specific editor. The data is provided in XML format. New topology is uploaded to the AutoBAHN Domain Manager after the Manager restarts.

The second topology used by the system is the inter-domain and it is called the Abstracted Topology model. It only contains reachability information and the attribute values are constant for a long period of time. This model does not contain technical details and its data is made available to the IDM module. This kind of topology is advertised to all the AutoBAHN domains in order to build a common inter-domain topology view. This advertisement is done by OSPF using an adapted Quagga routing engine.

The AutoBAHN system provides the dedicated module Pathfinder (inter-domain and intra-domain) which contains the algorithms and the logic to search for a path that satisfies each BoD reservation request according to specific sets of constraints, algorithms and policies. The module is composed of two main blocks devoted to inter-domain and intra-domain enquiries. The search returns a list of candidate paths using constraint-based shortest path algorithms. The AutoBAHN inter-domain path finding is performed according to the inter-domain abstracted topology (reachability information between domains + abstracted intra-domain links), which is built with an OSPF Quagga routing engine. Currently the Dijkstra algorithm is used to find the potential inter-domain path.

The information in the Abstracted Topology database is insufficient to perform a reservation; therefore an additional process (negotiation) is required to validate the path. During the process each domain is queried for the details and validation of the inter-domain part of the network path. The details are called constraints. The last domain on a path is responsible to validate all the constraints and prepare a final set of reservation attributes, which includes the technology specific parameters.

The AutoBAHN system only implements inter domain signalling based on RSVP. The Inter Domain Manager, which is responsible for initiating the resource booking, gets the reservation path from Pathfinder. In the next stage the reservation request is sent along the path. The communication is based on Web-Service technologies.

1.1.5 BLUEnet

BLUEnet is an intra-domain tool used to automate the provisioning of Ethernet point-to-point links over the HEAnet network. The control is centralized in the HEAnet NOC and the provisioning of L2 links is automatic; the user just has to select the two end points.

The BLUEnet tool uses two topology/interface discovery scripts written in *Perl* that run periodically every hour (*cron* job). For instance, there is a router script that logs into the topology database and looks for a list of all routers. The script then accesses each of these routers using IP (in-band) and works out the link adjacencies and checks what switches are connected to the routers. All this information is then stored back in the database.

In a similar fashion the switch Perl script logs into the topology database to retrieve a list of switches. It gets into each switch and checks several kinds of information: adjacencies, spanning tree information, VLAN information, etc. All this information is then stored in the database.

Routing of the MPLS traffic is done using IS-IS. In order for the config tool to find a circuit, the user via a GUI firstly selects the two end points and the type of circuit he/she wants (VLAN Mode or Port Mode). If the circuit request is confined to one spanning tree domain, the database topology is used to work out which switches need to be configured. If the circuit requested leaves one spanning tree domain, the exit point to the MPLS (Routed network) must be looked up. This information has been put into the database by the topology discovery tools. In this case an MPLS Layer 2 circuit is also built across the MPLS cloud to the other spanning tree domain.

The signalling of MPLS links is done automatically using LDP (Label Distribution Protocol). At the moment, Traffic Engineering is not used.

In summary, the Control Plane is a combination of spanning tree and IS-IS. Signalling is done using LDP in the MPLS cloud and topology discovery is done using a *cron* job that runs two *Perl* scripts and updates the database.

1.1.6 DRAC

DRAC classifies every domain according to its Control Plane. The tool has inter-domain routing capabilities and it accepts the management of centralized or distributed non-Control Plane optical devices. It works on an optical and packet switching architecture (L1 and L2) and it provides connectivity at Layer 3; however its architecture is composed of Layer 1 and 2 devices. The provisioning is automated using spanning tree for the topology discovery and PBT paths (disables STP) in order to reserve BW to this connection.

DRAC provides per VLAN bandwidth control, PBT setup, and dynamic layer 0/1 inter-layer routing. A layer 0/1 domain is abstracted into a single Layer 2 “virtual switching entity” for path computation across the Layer 2 network. UNI-N ports from the layer 0/1 domain are modelled as INNI ports on the virtual switch. Availability requests are issued to the layer 0/1 CPE/PCE to determine if a path through the virtual switch is available meeting the desired service criteria. If a viable Layer 2 path is computed using an INNI through the virtual switch, a reservation request is issued to the layer 0/1 CPE to reserve the bandwidth in the underlying network. Links through the virtual switch are modelled as dynamic point-to-point EVC with flexible bandwidth allocations.

Regarding QoS provisioning, DRAC creates virtual switches in order to accomplish the QoS parameters. It tries to get a path on layer 0/1 with these QoS requirements. If it is possible then it creates a virtual switch with this path. Finally, it tries to validate these QoS requirements at the edge of the virtual switch.

1.1.7 DRAGON

The DRAGON Control Plane architecture provides the missing components to make possible inter-domain end-to-end GMPLS transport. NARB (Network Aware Resource Broker) is an agent that represents a domain; it exchanges information with other NARBs representing other domains. This inter-domain exchange enables end-to-end LSP routing, so DRAGON provides end-to-end automated provisioning. The topology within a domain is discovered by listening to the local OSPF-TE protocol.

NARBs peer across domains and exchange topology information to enable inter-domain path computation and LSP provisioning. The NARBs utilize a modified version of OSPF-TE to share a link state database between domains. This inter-domain topology exchange can be based on the actual topology as discovered by listening to the local OSPF-TE protocol, or optionally based on an “abstracted” view of the domain topology (generated by configuration file or automatic synthesis of the OSPF-TE link state database). Domain abstraction provides mechanisms for an administrative domain to advertise to the outside world a highly simplified view of its topology. This allows domains to hide their real topologies as well as minimize the amount of external updates required. The trade-off is reduced accuracy for path computations. Each administrative domain can utilize configuration parameters to tailor its domain abstraction to the desired level.

In summary, the DRAGON architecture assumes that each LSR runs an intra-domain GMPLS routing protocol (e.g. GMPLS-OSPF). The NARB acts as a protocol listener to the intra-domain routing protocols and is also responsible for inter-domain routing. It also includes advanced algorithms which allow path computation with multiple constraints. These constraints include the standard GMPLS TE parameters as well as AAA, scheduling, multi-region switching capabilities, and vendor specific limitations such as switching capability adaptation abilities. The 3D Resource Computation Element (3D RCE) is the component inside the NARB that incorporates AAA and scheduling into path computation. 3D refers to TE constraints, AAA policy constraints, and time schedule constraints.

The output of the path computation is an Explicit Route Object (ERO) which may be a set of strict hops, loose hops, or a mixture of both. In addition, the NARB includes functionality to determine if multi-region and multi-domain techniques such as LSP nesting or stitching may be required. The goal is that once the NARB path computation is complete, signalling can progress with no need to return to an out of band path computation element.

NARB summarizes individual domain topology and advertises it globally using link-state routing protocol, generating an abstract topology. Then the 3D RCE computes partial paths by combining the abstract global topology and detailed local topology, and finally NARBs assemble the partial paths into a full path by speaking to one another across domains.

Each LSR runs an intra-domain GMPLS path signalling protocol (e.g. GMPLS-RSVP), and at provisioning time, based on the received ERO, the ESA (End System Agent) can initiate signalling via use of RSVP-TE or GMPLS UNI.

The inter-domain architecture and provisioning mechanisms are GMPLS based in the context that they rely on a link-state protocol for inter-domain topology exchanges, engage in multi-constraint path computation to determine appropriate network routes, and utilize RSVP-TE for signalling.

1.1.8 G3

G3 does not operate on the Control Plane.

1.1.9 GINS

GINS does not operate on the Control Plane.

1.1.10 MANTICORE/Argia

The infrastructure controlled/managed by the IaaS Framework is composed of devices from different domains and these devices can be interconnected, therefore, IaaS provides an inter-domain Control Plane which is distributed in order to assure that when a device fails, the network is not negatively compromised. The physical topology is not provided by the Control Plane because it is added manually using the Management Plane.

In MANTICORE, each router resource can be configured with static routes to another router resource or it can also be configured with dynamic protocols with their current parameters. RIP is used for internal routing. The following parameters can be configured: networks to announce, version (1 or 2), timeout, updating timers, metrics, garbage-collector timer, split-horizon enabling, static routes redistribution into announces, dynamic routes redistribution into announces.

OSPF is also used for internal routing. The following parameters can be configured: networks to announce and their netmasks, OSPF-areas, designated router/backup designated router, cost, hello interval, dead interval, static routes redistribution into announces, dynamic routes redistribution into announces.

BGP is used for external peering. The following parameters can be configured: Autonomous System number, BGP group names, BGP relationship (internal/external) for each group, announcement and reception routing policies, neighbours' names owning of each group and number of autonomous system of each neighbour.

Argia has a Bandwidth reservation service that allows to request point-to-point and point-to-multipoint connections. In order to calculate point-to-point end-to-end paths Argia can use two algorithms, the Dijkstra's shortest path and an algorithm that computes all the possible routes and lets the user to choose. Point-to-multipoint paths are computed either using an equivalent of Dijkstra's shortest path algorithm based on backtracking techniques or an algorithm that computes all the possible routes and lets the user choose.

In the IaaS Framework, the signalling functionality is based on Web-Services communication between the different components of these tools (between the end-user services like the Reservation Service and the IP Network Service and the network element controllers like the Optical Switch Service and the Router Service).

1.1.11 Juniper SRC

Juniper SRC supports the Policy and Management Plane and is connected to the Service Plane (Northbound interface) and the Transport Plane (Southbound interface). SRC supposes that all the inter-domain coordination is performed at the Service Plane layer (e.g. IPsphere SSS) with potentially activity at the Control Plane (e.g. BGP). The SRC's aim is to control the device of the operator domain, translating a service request into a policy in the Transport/Control Planes. The physical topology is not provided by the Control Plane because it is added manually using the Management Plane.

SRC exploits the routing protocols enabled by the network resources in the Transport Plane. SRC can import the routing table from JUNOS routers and use this information to identify the most specific match for IP addresses involved in the service activation so that policies can be applied to the right interfaces. It also exploits the signalling protocols enabled by resources in the Transport Plane, e.g. MPLS LSPs to allocate and reserve network resources and bandwidth end-to-end across the access, edge, and core network.

If a MPLS LSP setup is triggered through the SRC as part of a service then it can manage the state and bandwidth resources of that LSP. Optional resizing is supported.

1.1.12 PL-VINI

PL-VINI does not operate on the Control Plane.

1.2 Management Plane functionalities of the Tools/Frameworks

In this section all tools/frameworks are compared from the point of view of the OAM or Management Plane functionalities.

As in the previous section, it is of great interest to classify the Management Plane regarding their typology (intra- or inter-domain; and centralized or distributed). Other important features of a Management Plane are: managed resources (routers, switches, network nodes), resources discovery mechanisms (automatic/manual), failure recovery and protection (restoration, protection, alarms, failure recovery), performance monitoring, inter-domain policies, service provisioning, service management, security management (AAA, data confidentiality/integrity), and finally virtual management (resource partitioning, representation of virtual resources).

1.2.1 AMPS

AMPS is able to create a Premium IP service between two or more endpoints which belong to domains managed by AMPS. Resource discovery is manual and the configuration of routers for each domain is made during the deployment of AMPS.

A SLA module has been implemented and integrated, but no performance monitoring is involved. Management policy is implemented only on the intra-domain level.

In terms of security management, AMPS provides in each domain a dedicated Policy Module (PM) which allows the service provider to apply different policies to different groups of users. User group names have a hierarchical structure, and it is possible to apply general polices to the top level user groups, and/or more specific policies to specified user groups. Policies are determined according to the maximum and minimum duration of the request, the requested bandwidth and the total capacity that can be allocated for a group. In order to simplify policy management, the PM supports policy templates that can be used in order to apply the same policy to several groups. The PM also supports dynamic policies; i.e policies that are formed dynamically based on the values of certain parameters of the PM. Dynamic policies allow the modification of several policies by modifying the value of a single, common parameter.

Authentication in the AMPS is based on GN2-JRA5 eduGAIN. eduGAIN is an authentication and authorisation infrastructure (AAI) based on the confederation concept, and it provides the means to interconnect a set of national or community-wide federated AAIs. These participating federations cooperate to provide services to their member organisations and users beyond their limits. The confederation requires that both identity management and authentication/authorization services are properly handled by the participating federations, as it only provides the means to enable their interoperation. eduGAIN exchanges authentication data in SAML format and encrypts data with TLS standard. Each GEANT2 project can specify its eduGAIN meta data structure.

AMPS considers two possible scenarios of user authentication; the first one is a human user accessing the BoD service through a Web portal, so the user tries to access the AMPS application (IDM) in its domain through a Web interface. As this user is not yet authenticated, he/she is redirected by an eduGAIN filter to its local AA Infrastructure. Once the authentication is successful, the user can access the AMPS portal and make his/her reservation. The second one is an automated user accessing the AMPS application via a Web-Service. The automated user sends directly the authentication information to the AMPS application (IDM). An eduGAIN filter retrieves this information and sends it to the local AA Infrastructure. Once the authentication is successful, the user can access the AMPS application via a Web-Service and make his/her reservation.

Authorization has not been implemented yet, but it is a task in progress. In terms of accounting, actions of system user are not monitored, but the AMPS defines quota for each user which limits the number of reserved IP Premium services. Data confidentiality is ensured because all communication in the AMPS is secured by SSL protocol.

1.2.2 ANStool

ANStool does not handle connections in federated domains. It can only handle connections in a single domain. ANStool has a centralized topology mapping functionality with distributed control over the devices. The provisioning is done in a semi automatic way because the configuration is produced automatically but the administrators feed it to the corresponding network devices, until sufficient trust is gained.

Routers and switches are discovered by means of SNMP and manual resource discovery is implemented when there is no adequate support. In the ANStool there is no fault management, nor performance monitoring, neither service management. In terms of security management, four different types of users are defined: viewers, editors, administrators and network administrators. Viewers can not submit requests; they can only just view requests. Editors can apply for requests but can only view their own requests. Administrators can access to all the requests. Network administrators can view all the requests and they are able to grant them as well as to do some minor to them.

Authentication is performed in the framework of Shibboleth which is technology agnostic. It could be anything typically stored in an LDAP repository, but there is no collected accounting data. The aforementioned roles are coded in an attribute (eduPersonOrgUnitDN) variable accepting the previous defined values.

1.2.3 ARGON

ARGON is a centralized solution for automated provisioning of network resources within a single domain (intra-domain). However, ARGON can be integrated into a multi-domain environment by means of its Harmony/NSP interface developed in the EC-IST Phosphorus project. In this way, it would be possible to operate several domains, each controlled by an ARGON instance (or by another system providing the same inter-domain interface, i.e. Argia or DRAC).

Resources are manually assigned to ARGON. These resources are enriched with availability information throughout time, i.e. ARGON is aware of the bandwidth with respect to time. Restoration and protection are not supported, but path creation and termination are monitored and the path status can be queried.

The ARGON system, by itself, does not monitor network traffic. For IP services, traffic is shaped at the ingress such that SLAs (strict QoS) are met.

In a multi-domain environment, only the endpoints of the domain are disclosed to other domains and the availability of intra-domain paths can be queried. The source domain is in charge of finding a feasible inter-domain path (this is part of the Harmony/NSP extension) and does this by querying the availability of intra-domain paths within the different domains.

ARGON finds and establishes shortest paths given the constraints specified by the user. These constraints can be time constraints (start time, duration), bandwidth constraints (minimum bandwidth, maximum bandwidth), and delay constraints (maximum delay). ARGON also supports a mixed time/bandwidth constraint, the amount of data to be transferred by a certain deadline, with the “malleable reservation” type.

A reservation request can be comprised of several services (session concept) in order to apply for admission of a complex communication structure as a whole (instead of successively adding connections and then having to cancel all previous connections if one of them is denied). In case of rejections due to insufficient network resources at the requested time, an alternative (e.g. path with feasible time constraints) is replied to the user.

In a multi-domain environment, inter-domain paths are calculated by the Harmony/NSP code. It finds shortest inter-domain paths with respect to the number of domains traversed.

Services are managed automatically, as long as the user wants it. In this case, services are automatically activated at start time and terminated at the end time. Alternatively, the user can request manual activation; in this case, the service is not activated before an additional activation request is received. Also, services can be terminated before the requested end time by cancel requests. Service status can also be queried. SLAs are enforced by setting up dedicated paths, so SLAs are not met only due to failures leading to service cancellation by the system.

ARGON provides the possibility to identify users by username/password pairs for security management.

ARGON can be configured to manage a partition of the network topology. Partitioning and traffic separation between users is naturally achieved by reservation of dedicated paths for the connections.

1.2.4 AutoBAHN

The AutoBAHN system is managed by a dedicated AutoBAHN GUI. The AutoBAHN GUI is a Web portal which gathers information about accessibility of domain managers, allowing the request and cancellation of reservation services and shows a map with the registered domains. With this portal the user can also monitor the state of the submitted reservations. At this stage the management is centralized. Each AutoBAHN domain manager needs to register at the AutoBAHN GUI, and after registration the domain manager can be accessed by all users. All communication between the AutoBAHN GUI and the domain manager is made with the use of Web-Services technologies. Resource discovery is made manually during the configuration of an AutoBAHN domain manager.

AutoBAHN supports two types of automatic fault detection: the Control Plane fault detection and the Data Plane fault detection. In the Control Plane fault detection, the AutoBAHN domain managers monitor the work of peering domains. They can detect if the neighbour domain is down and inform all domain managers involved in the reservation service. In the Data Plane fault detection, the AutoBAHN provides a dedicated module for monitoring Ethernet network equipment, which can detect the malfunction of the physical infrastructure. In case of fault detection across the path of a reservation service, the old path will be removed and a new one will be established (not fully supported yet).

AutoBAHN also provides monitoring and managing of the accessibility of domain mangers from the GUI. AutoBAHN has three groups of users. The first group is “Projects”; there are a number of current projects that require high levels of bandwidth between different sites across Europe. Another group is “Connectivity to research infrastructures”. It has been identified that research users require access to a number of infrastructures for both their work and for collaboration. The last group is “Connectivity with special networks/sites”. Certain user communities have high bandwidth requirements for connectivity with some special networks/sites. The bandwidth needed by these three types of users is provided by an added value service such as BoD.

Authentication in AutoBAHN is based on the eduGAIN system. EduGAIN is an Authentication and Authorisation Infrastructure (AAI), based on the confederation concept. Accounting and policy issues are considered but they have not been implemented yet. There is a dedicated module to provide policy and authorization. It provides a user database with rules for accessing resources from the AutoBAHN system. It uses authentication data provided by the eduGAIN infrastructure and then maps this data with the policy and authorization model. AutoBAHN considers two possible scenarios of user authentication; the first being a user that tries to access the AutoBAHN application within its domain (IDM) via a Web interface, and the second scenario being an automated user accessing the AutoBAHN application via a Web-Service. In both scenarios, the user is redirected by an eduGAIN filter to its local AAI. Once the authentication is successful, the user/automated user can access the AutoBAHN Web portal/application and make the reservation.

Authorization has not been implemented yet. The GN2-JRA3 considers implementing a database or use the PERMIS environment. It will be based on user attributes (organisation, project, etc) and various kinds of parameters (maximum capacity allowed, maximum duration allowed, maximum delay, resilience).

All communication in the AutoBAHN system is secure by means of the SSL protocol.

1.2.5 BLUEnet

The BLUEnet tool has an intra-domain and centralized management.

As explained in the previous section (Control Plane), it uses two topology/interface discovery scripts written in *Perl* that run periodically every hour (*cron* job).

The tool has no recovery mechanisms, but in the Ethernet domain the network uses spanning tree to reroute around break points, while in the MPLS domain, MPLS will reroute automatically around fiber breaks, etc. Nagios monitors the up/down status of all the interfaces between end points. This is set up by the tool automatically for each new established circuit. Proactive monitoring of the HEAnet network and services availability is achieved through the generation of intelligent Nagios alarms.

The BLUEnet tool allows the establishment of “Port mode service” and “VLAN mode service” connections in an MPLS network. It configures Port mode/VLAN mode links over native Ethernet and L2 MPLS VLL clouds. The BLUEnet Web interface is used to manage these services, more specifically; it allows creating new L2 VPNs, listing active VPNs, VPN requests and deleted VPNs, and accessing monitored data from Nagios and Cricket.

BLUEnet has two different types of users: Admin users and End users. The long term vision of the tool is that End users will have control over a selection of interfaces on some devices and the setup and removal of circuits provisioned between interfaces. These interfaces need to be access-granted by the administrators. Admin users have setup and removal permissions over all circuits on all devices, and can grant these same rights to specific end users. In fact, an End user can create requests, but only an Admin user can push down the configurations and actually create the circuit.

BLUEnet uses a static dedicated database entry, with Admin users and End users, but in the HEAnet operational implementation the ANStool authentication method is not used anymore. Now *Pubcookie* is used, which is integrated with their centralized Cisco ACS (Access Control Server).

The implementation of the BLUEnet system is not yet finalised. Three specific areas need further development: (i) unfederated access to the tool by means of a static username and password, (ii) the partitioning of End users' circuits, and (iii) the creation of circuits by uploading the switch/router configuration to the active devices. This last task still requires manual approval by a NOC engineer, and therefore, it is not yet fully automated.

The Reservation/Activation/Deletion of a circuit is logged. A small feature that records all transactions that occur regarding a circuit has also been added to the system. The user ID is also recorded as a way of identifying who created/activated/deleted a circuit.

In summary, configuring circuits is done using a Web GUI (Web-Service on the way for machine to machine creation/deletion of circuits). The entire configuration is done in band using the config tool to push down the configuration changes. Nagios and Cricket are automatically created/deleted for each new circuit.

1.2.6 DRAC

DRAC has a single and centralized control domain. It provides network element abstraction and mediation services to DRAC components. It is based on existing OM/OMEA mediation solutions.

The DRAC Network Resource Broker provides inter-layer and inter-domain routing intelligence. DRAC has a Web-based service management GUI for end customer provisioning and monitoring of network bandwidth which provides graphical access to functions available through programmable user/network interface.

DRAC can follow the foot-prints from an application and from them, be sure that the network is giving the best behaviour for this application.

In terms of security management, DRAC has an external AAA support and provides integration into third-party authentication services.

1.2.7 DRAGON

DRAGON is a Control Plane architecture. In this sub-section the main DRAGON service, monitoring and AAA aspects are explained.

DRAGON allows the dynamic provisioning of network resources in order to establish deterministic paths (inter-domain end-to-end LSPs) in direct response to end-user requests.

Monitoring is not yet implemented in DRAGON, but DRAGONMon is an element under development that will monitor the status of existing LSPs and will store the data in a MySQL database to be accessed by other systems. A Web-based display of current provisioned circuits will be also included.

The objective of the policy-based provisioning is to incorporate AAA policy into path computation, resource allocation, and signalling functions. This requires high level associations of policy with users (or groups of users) as well as lower level associations of policy with actual network elements at a sufficient fidelity to implement meaningful policy based resource allocations. For the higher level

AAA functions, there is related work underway or completed in several communities (e.g. GSI by the Globus consortium and Shibboleth by Internet2).

What is missing from much of this work is an extensible and scalable architecture to translate these higher level associations to the network provisioning level. As a result, DRAGON is focused on the development of architectures and mechanisms to translate the high level AAA information into information which can be utilized directly in the provisioning process. The current approach is the 3D RCM (Resource Computation Model). In this model, the higher level AAA information is synthesised into policy information which is associated at the TE resource level. Also, the higher level AAA information is used to develop a set of policy rules. The TE policy data and the policy rules are used during path computation to incorporate AAA policy into provisioning operations. In so doing, AAA policy decision can be synergized with TE based provisioning decision, resulting in fast, precise and simplified control process.

1.2.8 G3

G3, from the SNMP based monitoring point of view, does not know anything about logical grouping nor administrative borders in the network, it just needs connectivity to devices that shall be measured and read-only access to appropriate SNMP data trees. Measured devices can be organized into logical administrative groups at the UI output level (per device configuration) but this is rather for visualisation purposes than a consolidation of the measurement data.

The measurement of the FEDERICA base infrastructure will be centralized and will be done from CESNET using a G3 instance dedicated for FEDERICA.

G3 measured data is currently stored in a RRD database. The UI needs to access data at file level, and when operating in a distributed architecture (more measurement instances and a common UI), data from measurement instances is provided using read-only NFS.

There is no method for automatically discovering new devices attached to the network, but there is a device reconfiguration tracking. During “discovery” (or first measurement of a device), an automated search is done and the result of this gives us items, that can be currently measured and all instances where they occur.

The G3 system is designed to allow the implementation of alarms (i.e. an alarm could be generated during measurement), but this has not been set up yet.

Another way for generating alarms can be using the report preparation information, for example when generating the "network health map".

Some of the key features of the G3 System User Interface are:

- Everything that the G3 System UI does refers to a time window, which is defined on the basis of ”From” to “now”.
- Each network device is usually perceived as a logical structure consisting of several groups of objects and navigation is done with the help of a tree-like structure consisting of a hierarchy and lists of such identifiers or items. This tree is the structure of objects to measure.
- The time step of measurements changes dynamically. The configuration is device based and consists of a sequence (unlimited in size) of time steps where each time step can be set up as exact or semi-random time interval. It gets relatively short time peaks of values and it is less aggressive.

- Aggregated views
    - Sums, minimums, maximums of aggregated values.

The G3 system has two basic types of users, network administrators and end users. Network administrators use the G3 system interactive interface. This interactive interface is designed for device structure browsing, searching for objects and for the visualization of what was selected.

End users do not use the interactive UI. They have access to static reports generated by the G3-reporter (a standalone tool working above the interactive user interface). Report content is given by specific configuration; an example is a report about CESNET2 IP/MPLS backbone utilization (link, capacity, bit rate, utilization).

Interactive UI authentication relies on an HTTP server. Access control to specific measured data could be implemented, but is currently file-based. Access to generated reports is given by the HTTP server.

Monitoring of virtual resources is not implemented, and the difficulty to adapt G3 to do it depends on vendor implementations and on the measurement architecture. It will be tried to measure virtual elements from both outside and from inside. From inside they should look like common physical devices. From outside there will probably be more information available but a lot of vendor proprietary OIDs to adapt. G3 does not work with real OIDs, it works with its own internal virtual items which may (or may not) have a close relation to real OIDs.

1.2.9 GINS

GINS presents a centralized management of the available functionalities and is able to handle intra-domain monitoring and to support inter-domain monitoring compliant with the perfSONAR framework and NM-WG.

Resource discovery is not supported, so the monitored devices and services must be manually configured in the GINS DB.

GINS provides fault detection, notifications and reporting. Regarding fault detection, GINS detects and defines the status of the monitored interfaces, protocols and services, e.g., user connectivity, lambda and E2E services, network equipment. In particular, it performs fault detection on interfaces (physical and IPv4 and IPv6 reachability status), lambda services, Sonet/SDH sections, MPLS LSPs, multicast beacons and BGP sessions. Fault detection is performed on the basis of SNMP polls and ping responses. Performance monitoring data and alarm triggers are based on customized data correlation algorithms.

Fault notification is done by alarms in the GUI, emails and Trouble Tickets. GINS provides fault reports, such as circuit fault report, Trouble Ticket, alarm history, Sonet/SDH error and physical interface input error or output drops statistics.

The main functionalities of GINS are classified in 4 groups:

- Monitoring functionalities: GINS detects/defines the status of different services, on the basis of the information gathered through the network. Monitoring is supported on the following service classes:
    - IPv4 and IPv6: end-user sites and backbone interfaces [service status, input errors and output drops on physical interfaces]

- IP Multicast Beacons [service status]
- Routing protocols: OSPF [link costs] and BGP [peering status, advertised routes]
- Sonet /SDH router interfaces relative to leased-lines [SDH/Sonet errors]
- Lambda [service status, optical equipment port status]
- MPLS [MPLS LSP status]
- E2E: each E2E service is defined as the stitching of multiple intra-domain and inter-domain links. The status of the E2E service is defined on the basis of the status of all the intra/inter-domain links provided by each domain. Furthermore, each domain link is made up by an ensemble of heterogeneous links, e.g., IP links, MPLS LSPs, lambda, etc. GINS is able to define the status of domain links on the basis of the status of the constituting links and to export this data to the GN2 E2E Monitoring System [E2E service status]

- Statistics functionalities: GINS stores performance measurements data, e.g., statistics relative to IP bandwidth usage, Sonet/SDH errors, etc., and provides graph (as a function of time) on the following metrics:
  - IPv4 and IPv6 Traffic Statistics: backbone link traffic statistics, user traffic statistics, aggregate traffic statistics, peering traffic statistics, backbone traffic weathermap, premium IP traffic statistics.
  - Sonet/SDH error statistics
  - Router CPU load statistics
  - End-user Uncompressed Statistics, i.e. 5 minute bandwidth usage data since the circuit setup
- Fault and performance reports functionalities: GINS provides the following network fault and performance reports:
  - User Monthly Report (HTML and PDF): user fault report and circuit availability, uncompressed traffic statistics (IP Bandwidth usage, 95th percentile).
  - Carrier fault report and circuit availability (HTML and PDF)
- Data export functionalities: XML data export has been developed in order to support perfSONAR and GN2 E2E monitoring service and according to compliant data format.

GINS is compliant with the perfSONAR and NM-WG inter-domain monitoring policies. In particular, GINS is able to provide E2E service status, link utilization and input errors/output drops data export.

GINS includes a set of mechanisms aimed to diagnose problems in the monitoring service process. In particular, it is able to automatically perform monitoring service failure notification (problems in PM data gathering), to check data consistency (verify consistency between GINS configuration files and the network), and to detect misconfigurations or errors in the layer/technology stitching process.

In terms of security management, GINS uses two user authentication mechanisms, the former is based on NIS authentication (used to manage GARR internal credentials), while the latter is based on MySQL which stores the external user credentials. Afterwards authentication, each user is assigned to one group.

In GINS every frontend or backend element, data or application is associated with a specific task, and each task is defined by a set of permissions for groups and users. There are only Apache log messages available in order to record user access. GINS allows accepting or denying an action for a specific user or user groups.

Data confidentiality is achieved in the GUI via HTTPS data encryption.

In principle, GINS does not have slicing capabilities and representation of virtualized resources, but the GUI is able to represent different data relative to different network slices on the basis of the user credentials (and of course of the equipment attribute configured in GINS).

1.2.10 IaaS Framework

The IaaS Framework has an intra-domain/inter-domain and distributed management. In the former, infrastructure resources can be traded between domains either by using a direct exchange or well known resource brokers. End-user services deployed on top of the infrastructure resources can be intra-domain or inter-domain depending on the implementation. In the latter, distributed management, each type of resource (switch, router, optical device) is managed by a Web-Service and each device is represented as a resource (using WSRF). Different or the same Web-Service applications can be deployed in one or several machines and WS resources can be distributed between several of these machines.

Physical resources discovery is triggered through the graphical interface. The physical administrator user has to introduce the name of the device, the owner, administrator user ID, type of transport, router configuration IP address, configuration port, configuration protocol, login and router password, device model, connection information and security through an intermediate host and other additional options. Then the system automatically queries the current device in order to obtain the inventory (physical ports) and the response is stored with persistence.

In the IaaS Framework, if any operation fails, an exception is thrown by the “Device Virtualization WS” and two possible actions can be performed according to different faults. The first is to recover the last action (i.e. undo the last operation), and (perhaps) sending a new operation to the device). The second is to send an exception to the upper layers, so that this alarm is shown at the Graphical User Interface.

In order to operate end-to-end across multiple independently managed domains, MANTICORE administrators (who administer this resources from different domains) can configure external peerings. The protocol used for this is BGP and the administrator can introduce several parameters for this protocol, as a “policy”.

In the IaaS Framework, services can be activated through the GUI or with an application that calls the WS API. Argia can detect and signal service overrides or misconfigurations. The conflicting resources are marked as inconsistent and some informational actions like displaying the problems in the GUI are triggered. Alarm management and service misconfiguration detection are not implemented yet in MANTICORE.

In terms of security, there are three different kinds of users with different Use Cases, Physical Network Administrators, APN or Virtual Network Administrators and End Users. The Physical Network Administrator is the owner of the physical infrastructure. He/She can virtualize (create a software object that represents all or a part of a physical infrastructure) the physical infrastructure and give permissions to users (export virtual resources) so they can have limited control over the resources. The APN gathers resources from one or more physical network administrators or other APN Administrators (by asking them or through resource brokers). He/She also assigns the resources he/she has harvested to different services (for instance, he/she could assign a set of Ethernet port resources to the VLAN Service, or a set of logical routers to the IP Network Service). Finally, the End User just uses the services provided by the software. For instance, if there is a reservation service, the user can ask for network reservations; if there is an IP Network Service, the user can configure its IP network; if there is a VLAN Service, the user can ask for an end to end VLAN, and so on. This type of user just sees the service interface; he/she cannot collect or trade virtual resources, so it is the traditional end user.

In order to access to the GUI, all users have to start a session according to their role. An authentication dialogue asks for a username and the password. For each kind of user and according to his/her role, different services and functionalities are showed to him. Each user has his/her profile and workspace in the server and all information related with his/her owning resources is stored in memory and in the database. Current resources, networks and maps are downloaded from workspace at session start and uploaded to it when some changes are done.

Data integrity and confidentiality is assured because each user/admin selects a set of resources from his/her workspace (only resources that he/she is allowed to administrate) and from that time, these resources become locked for the other users until he/she finishes using them, so information about them can only be viewed and modified by one user.

One of the main capabilities of the services that use the IaaS Framework is virtualization. Using MANTICORE, the physical administrator can create or delete several logical routers from one Juniper router. Each of this logical router or also the physical router can be virtualized (abstracted) and router interfaces are distributed in these Logical Routers. This kind of users can also create and delete Logical Interfaces from a physical Interface. Each of these logical interfaces can also be assigned to one logical router. Using Argia, optical devices can also be partitioned creating several logical ports (i.e. a subset of TDM channels).

Another concept of slicing is possible with services that use the IaaS Framework. Physical and logical resources are represented by a software instance that user can use in order to orchestrate his/her own network. So, a virtual slicing of the Physical infrastructure in several virtual infrastructures is possible.

In order to allow users to manage physical devices or partitions of them, a software layer is used. Each physical device can be abstracted with a software entity that represents it and works as its driver. Each of these software entities is a Web-service resource controlled by a specific kind of WS (router WS, optical card WS, Ethernet WS) using the WSRF (Web-Service Resource Framework). When a service requires to configure or to get information from a physical or logical (a partition of it) device, it has to address to its software resource.

In order to get a better management of the resources, Physical administrator groups them (after abstracting them with software instances) using Resource lists (a set of virtualized resources). These resource lists can be sub-leased (exporting them) to other users. After sub-leasing them, the first user cannot use these resources because only the new target user can use them. From that moment, this administrator user can reorganize his/her resources in other resource lists, sub-lease them to other users, or use them in order to create his/her own Virtual Networks.

1.2.11 Juniper SRC

The SRC software supports four types of services:

- Normal (policy-based service)
- Aggregate (group of services, handled as a unit)
- Infrastructure (service that can be provisioned only once and then activated a number of times for one or more subscribers across network devices)
- Script (custom service into which you can insert or reference a script that provisions policies on a number of systems across a network, including networks that contain devices that do not have supported device drivers).

Aggregate and infrastructure services are used together to apply policies across JUNOS routers, JUNOS routing platforms, and other systems that have a supported device driver. Script services are used to create customized service implementations, such as a service to configure firewall policies on a device that does not have a supported device driver.

The provisioning of services can be accomplished via CLI, a Web interface or Netconf (XML). Services and policies allow parameterization for enhanced flexibility and scaling.

The SRC system allows the monitoring of active services per user/interface. If additional information about the service definition is needed it can be looked up as well. This applies to the CLI and the Web interface. Via Netconf the same options are available but the graphical representation is done by a tool outside the Juniper SRC. Statistics and SNMP traps can be used to monitor the systems and detect failures/bottlenecks. Log files for the various modules within SRC can be configured with different debug levels and destinations. As destinations a local file or a syslog server are possible.

Concerning security management, the SRC provides fine-grained access controls for administrators of the system, including remote authentication mechanisms such as TACACS and RADIUS. This applies to system administration via CLI, WEB or NETCONF.

For services enabled by SRC on JUNOS routers, accounting information is collected from the router and can be exported into a flat file or RADIUS accounting messages. The information provided in those can be customized to the requirements, and additionally, an open API can be used to integrate it with other back office systems.

In terms of virtual management, generic service templates can be invoked by different resources. Each requester might add an identifier used to uniquely identify the service instantiated based on the request. This can be reflected in the accounting details of a service.

In the case that resources are monitored by the Admission Control Plugin (ACP), then the ACP takes over the responsibility to verify that resources are available.

1.2.12 PL-VINI

PL-VINI works as in a single domain but with resources distributed over the Internet. The control is central in the sense that there is only one controller and the provisioning is automatic across the various PL-VINI nodes which are identified manually.

There is no restoration or/and protection. Alarms about failures are carried on a central monitoring tool.

There are various roles for PL-VINI nodes. Local administrators may run local experiments as they like, non local users are added by means of their SSH keys and PL-VINI administrators are super users.
Authentication is done via SSH, and there is no authorization because each user is assigned his/her own slice.

There is performance monitoring for accounting data produced from different slices. Basically an application called "CoMon" counts the number of packets, number of bytes and cpu usage.

The PL-VINI manages resources partitioning using the Vserver virtualization software and the VNET socket for network isolation between slices. Each node is able to host several slices and the only orchestration of resources is available is through Emulab remote procedure calls (RPC).

1.3 Network services controlled by the Tools/Frameworks

In this sub-section, the tools/frameworks are compared from the point of view of the Transport or Data Plane. The supported network technologies are identified and also how is modelled the stitching between them. The type of physical devices that can be controlled/managed/monitored by the tools will also be compared, and finally the kind of connectivity that provides each of the tools is identified.

- AMPS
  - Supports Layer 3 only (Premium IP Service)
  - The software is claimed to work properly with all kinds of Juniper and Cisco L3 devices.
- ANStool
  - L1/L2 circuits (Lambdas / SDH / OTN), Ethernet (VLANs, Q-in-Q, PBB-TE MAC in MAC), MPLS (VPLS, VPN L2, VPN L3).
  - ANStool supports Cisco Layer 2 and 3 devices, Extreme Layer 2 devices, any SNMP capable device, any device which can be managed efficiently by CLI.
- ARGON
  - ARGON provides:
    - Layer 2: End-to-end Ethernet tunnels via MPLS/VPLS (Riverstone), end-to-end Ethernet tunnels via GMPLS (Alcatel), VLAN and port configuration (Riverstone) and traffic shaping, i.e. bandwidth restrictions for end-to-end connections, resulting in support of QoS.
    - Layer 3: End-to-end IP tunnels via MPLS (Riverstone), mapping of IP packets to MPLS tunnels (route maps, Riverstone), traffic shaping, i.e. bandwidth restrictions for end-to-end connections, resulting in support of QoS.
    - ARGON provides its own path computation entity, i.e. MPLS and GMPLS paths are computed (also reserved and provisioned) to provide traffic engineering capabilities with respect to the time dimension.

- It supports Riverstone 15008 and Alcatel 1678.

- AutoBAHN
  - The GEANT AutoBAHN system is focused to the following technologies: Layer 1 (lambda circuit management), Layer 2 (MPLS VPNs, Ethernet, SDH, GFP over SDH)
  - AutoBAHN has been integrated with the following network equipment: the Alcatel NMS ISN interface as a technology proxy for the GEANT2 testbed, the GRNET ANSTool as a technology proxy for the GRNET backbone, the HEANet BLUEnet tool as a technology proxy for the HEAnet backbone and the PSNC tool for configuring VLL circuits over an Ethernet infrastructure
- BLUEnet
  - The BLUEnet tool offers two type of L2 services:
    - PORT MODE: configuration of 802.1q ports to perform Q-in-Q between the customer and the HEAnet owned CPE (Customer Premises Equipment), traffic transported transparently across the MPLS network over pseudowires.
    - VLAN MODE: presented to the user as an 802.1q trunk, each customer VLAN is mapped to an MPLS LSP (VLAN id mapped to a EoMPLS VCs)
  - Cisco Catalyst 3750 Switches, Cisco ME 3400 Ethernet Access Switches, Cisco 7600 Routers
- DRAC
  - Provides basically VLANs.
  - Supports a Layer 2 switched network where end clients reside and a layer 0/1 switched optical network with Layer 2 INNI connectivity points, or Layer 1 UNI.
- DRAGON
  - Handles Layer 1 (WDM), Layer 2 (LSPs), Layer 3 (routing and path signalling)
  - Controls switching components that have their own native GMPLS protocols, and there is the Virtual Label Switch Router (VLSR) that allows end-to-end automated provisioning including switching components that do not have their own native GMPLS protocols (Ethernet, TDM and Optical switches).
- G3
  - G3 was designed for a multi-vendor and multi-layer network environment. Although it is open from the data collecting point of view, main data collecting method is expected to be SNMP (internal support, currently based on the *libsnmpsession* package). Open issues relate to the supported MIBs, OIDs and the ease of adding new OIDs from new MIBs. G3 is being used to measure the whole CESNET2 network (including its DWDM infrastructure) and within the widely acceptable scope (SNMP related RFCs) it is able to measure any device (tested or continuously measured are L2, L3, optical Cisco boxes, Force10 and Foundry switches, linux boxes [net-snmp based agents], Juniper boxes and others).

- GINS
    - GINS operates on the following layers and technologies: lambda, Sonet/SDH, MPLS (L2 and L3 MPLS VPN), E2E circuit, IPv4, IPv6 and IP Multicast, IP Routing protocols (BGP, OSPF).
    - GINS is currently used to monitor the following equipment: Juniper (J6350, M7i, M10, M20 and M320), Cisco routers (17xx, 18xx, 2xxx, 3750, 72xx, 75xxm and 12xxx), ADVA FSP3000, Metrobility (R4000, R5000).
- IaaS Framework
    - Several network technologies can be supported by tools that use *IaaS Framework*. Up to now, with MANTICORE and Argia, L3-IP and L1/L2 circuits respectively can be supported. With the deployment of new Tools using *IaaS Framework* new technologies as Ethernet (VLANs), MPLS could be supported.
    - MANTICORE supports routers and routers with logical routing capabilities and Argia supports optical switches.
- Juniper SRC
    - The Juniper SRC comes with two onboard copper 10/100/1000 Mbit interfaces and provides two optional SFPs for either copper or optical Ethernet modules.
    - Out of the box there is support for Juniper J, M, MX, T, TX series via XML and E-series via COPS protocol. Other options which would require some adaptation are based on RADIUS CoA or a script API which allows to use Telnet or other methods to configure a network devices.
- PL-VINI
    - Supports this type of connectivity: GRE tunnels, UDP tunnels and RAW SOCKET.
    - PL-VINI manages virtualization of PCs and the routing and switching software inside slices.

1.4 Services offered by the Tools/Frameworks

In this section the main services that each tool tools/frameworks offers are exposed.

- AMPS
    - AMPS enables authorized end-users to make a single reservation for Premium IP bandwidth (that is to say, a guaranteed uncongested path) that is effective across a chain of participating domains.
- ANStool
    - The objective of the tool is to provide Web-based application for unidirectional and bidirectional bandwidth reservations optionally inside a Layer 2 or Layer 3 VPN.

- ARGON
  - ARGON supports six different services to provide network resources to users. These services are based on MPLS and GMPLS networks: reservation request, availability request, cancel request, modification request, activation request, binding request
- AutoBAHN
  - AutoBAHN is a bandwidth reservation system and signalling interfaces that allow end-users to make advance reservations (inter-domain) with automated provisioning.
- BLUEnet
  - The BLUEnet tool permits the establishment of “Port mode service” and “Vlan mode service” connections in an MPLS network. It configures Port mode/Vlan mode links over native Ethernet and L2 MPLS VLL clouds. The BLUEnet Web interface allows to create new L2 VPNs, list active VPNs, list VPN requests, list deleted VPNs, and access monitoring data from Nagios and Cricket.
- DRAC
  - DRAC provides connection between different hosts used by a grid application with some QoS parameters, a Web-based service management GUI is used for end customer provisioning and monitoring of network bandwidth.
- DRAGON
  - DRAGON allows dynamic provisioning of network resources in order to establish deterministic paths (generally a set of LSPs) in direct response to end-user requests.
- G3
  - G3 system service is basic to monitor data in a network infrastructure and allow access to the data by the users, there are to ways of accessing the results: interactive UI and generated reports (Reporter).
- GINS
  - Network service monitoring (monitoring, statistics, trouble ticket system, fault and performance reports)
- IaaS Framework
  - Up to now, connectivity services as the “Administration of IP Networks” (configuring internal Routing protocols, and external peering capabilities) with MANTICORE and creating optical paths with Argia are available. Services can be activated though the GUI or with an application that calls the WS API.
- Juniper SRC
  - The SRC Portfolio supports a suite of applications that enable the creation of a myriad of services, as for example: Multiplay Services (e.g. Bandwidth on Demand), Tiered Access Services and Subscriber Self Provisioning, Enhanced Security Services, Multiplay Services with IMS and non-IMS applications. The SRC software

supports four types of services: Normal (policy-based service), Aggregate (group of services, handled as a unit), Infrastructure (service that can be provisioned only once and then activated a number of times for one or more subscribers across network devices), Script (custom service into which you can insert or reference a script that provisions policies on a number of systems across a network). The provisioning of Services can be accomplished via CLI, a Web interface or NETCONF (XML).

- PL-VINI
  - The tool is able to build virtualized topologies on top of the Internet as an overlay.

1.5 Advantages and disadvantages of each Tool/Framework for the FEDERICA approach

In this section, the main advantages and disadvantages of each tool/framework for the FEDERICA approach are listed. The aim of this section is to help in the selection of the tool that best fulfils the FEDERICA requirements.

1.5.1 AMPS

Advantages

- Fully functional inter-domain and intra-domain reservation system for Premium IP service.
- Flexible user and group policy modelling.
- Flexible modes of working (Manual, Semiautomatic, Automatic).

Disadvantages

- Support only for Layer 3.

1.5.2 ANStool

Advantages

- The tool has been developed around the GrnetDBII using PHP.
- Dynamic generation of tool configuration files.
- High number of monitoring metrics on different layers.
- Highly customizable.

Disadvantages

- The tool is tightly coupled with the modelling of the network namely GrnetDBII. GrnetDBII models the topology and the resources of the network using an Object Oriented implemented in OOPHP. The tool has not been documented thoroughly although it is written in PHP. It has not been transferred to sourceforge.

1.5.3 ARGON

Advantages

- Support of advance reservations.
- Support of point-to-point and flexible data transfer services.
- Support for reservation requests containing multiple services.
- Availability requests and alternatives to support Grid scheduling and co-allocation.
- Services are provided via Web-Services (e.g. used by Grid schedulers) and via a graphical frontend.
- Interoperability with the multi-domain solution of Phosphorus Harmony / “Network Service Plane” (NSP). Harmony is implemented and tested in the IST project Phosphorus and interacts with DRAC, UCLP / Argia / IaaS framework, and ARGON.
- Due the internal architecture of ARGON, new modules for unsupported routers can easily be integrated.
- System is proved to be functional in the VIOLA testbed and was shown at several demos.

Disadvantages

- Restricted set of supported hardware requires development of new modules for additional hardware support (will be carried out by University of Bonn).
- System not yet in productive state, limited support by University of Bonn.

1.5.4 AutoBAHN

Advantages

- Multi-domain approach for bandwidth on demand reservations.
- Physical topology modelling.
- Flexible technology adjustment (Ethernet, SDH, VLAN, MPLS, etc.).
- Advance reservation scheduling.
- Easy, well-documented deployment.
- Flexible related tools.

Disadvantages

- Not in production, still in development phase.
- Some less probable exceptional situations may not be served properly (service may be denied to user).

1.5.5 BLUEnet

Advantages

- Easy to use by a researcher without specialist training.
- Speed of creating/deleting p2p links: minutes.

- Consistent configurations (and monitoring) setup.
- Single point for looking up circuit information. Querying circuit information.
- Alarm monitoring and graphing tools automatically configured for each new circuit.
- Auto-discovery of new switches added to the network.

Disadvantages

- The BLUEnet tool provisioning is limited to MPLS L2 circuits.
- The BLUEnet tool is limited to a single management domain (HEAnet network).
- The BLUEnet tool is immature in their development and currently not integrated into any grid middleware.

1.5.6 DRAC

Advantages

- Can create paths at L2 with QoS requirements.
- Can create virtual switches from L0/1 connections.

Disadvantages

- Cannot create slices from physical devices.
- Cannot operate above L2.
- Cannot manage PCs and VMs.
- Cannot manage switches or routers.

1.5.7 DRAGON

Advantages

- DRAGON is multi-domain.
- Allows provisioning across multi-domain with AAA and scheduling features.
- Much software seems to be available and operational.
- Open source and source code accessible.

Disadvantages

- DRAGON targets specifically GMPLS.

1.5.8 G3

Advantages

- Flexibility and extensibility in terms of adding new OIDs and OID sub-trees in cases when new group of information will be needed or new vendor will be added.
- Fully configurable strategy of measurement time step (time step is expected to be dynamic).

- Automated periodical device item discovery (non-aggressive) and device reconfiguration tracking.
- Independence from SNMP instance identifiers (generated by devices - like *ifIndex* and similar).
- Possibility of information unification in multi-vendor environment, any real OID set (pointing to the same information on different vendor devices) may be hidden behind internal virtual identifier (abstraction may be recursive).

  Flexible output report generator (Reporter) based on interactive UI session management, it can give overall views (various type of reports utilizations, health, multicast, others) on the whole FEDERICA infrastructure.

Disadvantages

- It is SNMP based monitoring system.
  - Nobody can expect micro-time perspective.
  - Can provide SNMP measurable information only.

1.5.9 GINS

Advantages

- Integration of many monitoring tools.
- Dynamic generation of tool configuration files.
- High number of monitoring metrics on different layers.
- Highly customizable.

Disadvantages

- Highly customized on GARR technical and administrative operations.
- Poor software documentation.
- PC hardware and software (services and applications) monitoring not currently developed (but supported).
- Official software package currently not released.

1.5.10 IaaS Framework

Advantages

- Can be adapted to the IPsphere Business Plan.
- It offers common functionalities that can be used for several types of networks and devices. Therefore, it offers huge modularity and scalability.
- Different permissions for different types of users.
- Transparency for the upper layers when configuring the devices.

- A set of libraries are been implemented in order to configure/monitor (communication) different physical devices. Three main components of this library are transport components, protocol components and command components.

Disadvantages

- Monitoring Services are not implemented (but new services for it can be adapted to the framework).
- GUI must be enhanced in order to operate with other services.

1.5.11 Juniper SRC

Advantages

- SRC software modules utilize widely adopted standards-based open interfaces to maximize interoperability with the broadest range of elements, applications, and platforms.
- Rich fully integrated with Juniper Networks´ routing platforms (including those used in FEDERICA) and security solutions, and extend support to third-party network elements.
- The SRC software modules feature a powerful meta-directory capability for rapid integration with external repositories and related systems that support billing, customer care, order entry, provisioning, billing and security.
- SRC gateway modules utilize standards-based interfaces to integrate with a broad set of service layer applications, enabling application driven control of network resources:
  - SRC Diameter Gateway (SRC-DG): The SRC-DG provides a Diameter “northbound” interface that enables the SRC-PE to perform the Access Resource and Admission Control Function (A-RACF) and to provision policies for Session Initiation Protocol (SIP) services across the IP/MPLS network infrastructure.
  - SOAP Gateway (SRC-SG): The SRC-SG uses the Simple Object Access Protocol (SOAP) to provide a simple, straightforward and standard-based interface that allows a non-SRC application to integrate with and leverage SRC Software Modules. For example:
    - Integration with Grid applications.
    - Integration with FEDERICA inter-domain service middleware.
- Carrier-class solution: Easily dimensioned solution, with predictable performance and scale.
- Pre-packaged solution: Provides a solution that is easy to dimension, test, install, configure, upgrade, troubleshoot and spare. It simplifies the operations environment.
- Flexible Service Activation: Provides mass service customization, stronger end-user relationships, and competitive differentiation without increasing operational complexity or costs.

Disadvantages

- Provides only part of the solution, i.e. the Policy and Control Management Plane (not the Service Plane).
- As with any new tool, a learning phase needs to be considered.
- Intra-domain only. However it provides an interface to be connected to a FEDERICA specific service layer which would take care of the Inter-Provider offering.

1.5.12 PL-VINI

Advantages

- The PL-VINI ingredients are based on open source components. PL VINI builds upon the successful tradition of Planetlab. The tool is able to build virtualized topologies on top of the Internet as an overlay. It is possible to run the Internet in a slice. In this mode the user can connect his/her slice to the Internet.

Disadvantages

- The “controller” is a monolithic software in a closed source format which does not leave enough space for customization. The tool does not work in a federated approach. It does not produce network centric topologies but rather overlay topologies.

1.6 Licence of the Tools/Frameworks

In this sub-section the type of licence of each tool/framework is explained in order to have an idea of the permissions we will need if we decide to use and/or enhance them in the FEDERICA project.

The tools/frameworks that have an open source licence are ANStool, DRAGON, PL-VINI. MANTICORE is based one Apache Software Licence (ASL).

Regarding GINS, in principle the software is also open source, however GINS has not been officially released and details on the licensing policy have not been yet defined.

AMPS and AutoBAHN are being developed by the GEANT2 project, and it has not decided yet, what kind of licence they will apply, but they are considering BSD, Apache Foundation or GPL, so they will be open source and we will be able to use them in FEDERICA.

ARGON, G3 and BLUEnet are tools/frameworks that can be provided for use within the FEDERICA project, free of charge.

Finally, there are two tools/frameworks with fully commercial licences: Juniper SRC and DRAC. Juniper SRC has a demo licence, but this is limited in time.

2 Comparison of the Tools/Frameworks functionalities with the main FEDERICA requirements

2.1 Operational requirements:

- Slice isolation in a multi-user environment
  - GINS: It is able to isolate monitoring data relative to different network slices on the basis of user credentials.
  - IaaS Framework: Several users can use the system at the same time. If a user is setting some resources, these resources are blocked for the control of other users.
  - PL-VINI: Complete isolation through Vserver and VNET.

- Inter-slice communication possibility
  - GINS: PM data relative to different slice can be correlated and/or grouped according to the user requirements.
  - IaaS Framework: Connectivity with other infrastructures is possible configuring external peering in specific interfaces.
  - PL-VINI: Yes, possible via UML switch.

- Inter-connection with other infrastructures
  - ARGON: IST Phosphorus Harmony / NSP.
  - AutoBAHN: DRAGON, Alcatel NMS, HEANet BLUEnet, Grnet ANSTool.
  - DRAGON: Allows peering with other infrastructures.
  - GINS: It is currently interconnected with the perfSONAR system. The available data export functionalities could be customized in order to communicate with other monitoring systems.
  - IaaS Framework: Resources can be exported to other infrastructures and vice versa.
  - PL-VINI: Possible by using openvpn.

- Connection to Internet
  - AutoBAHN: The AutoBAHN supports both private and public network.
  - GINS: Supports monitoring in both private and public network.
  - IaaS Frame: IP parameters can be configured on the Resources, so the Internet connection parameters can be configured in order to go out to the Internet.
  - PL-VINI: Yes, through NAT.

- Apply changes to network manually
  - AMPS: Works in 3 modes: manual, semi-automatic and automatic.
  - ANStool: It supports such type of operation.

- AutoBAHN: System allows automatic applying changes to network equipment.
- BLUEnet: It supports such type of operation.
- DRAC: It offers full control of the devices.
- GINS: Device and service monitoring can be configured (setup or changed) online manually.
- IaaS framework: Depending on the kind of change.

- Modify the slice after the original request
  - ANStool: Yes. Although there is not any slice concept, you can modify a VPN after creation.
  - ARGON: Modification of the set of reserved network resources is possible.
  - GINS: Network slice monitoring can support changes on slices.
  - IaaS Framework: Admin users can modify their slices after its creation. They can add or drop resources from their own networks.
  - PL-VINI: Possible after request.

- Repeatability of experiments
  - AMPS: The AMPS allows repeatable reservation of Premium IP service between all control endpoints.
  - AutoBAHN: The AutoBAHN system allows requests services as often as needed. The only constraint is physical network infrastructure.
  - IaaS Framework: The software can configure identical scenarios over different periods of time. The reproducibility of the experiments depends on the behaviour of the Network Elements.

2.2 End user requirements:

- Sub-lease of virtual resources to end user
  - AutoBAHN: The AutoBAHN provides the bandwidth or circuits for the user across different domain.
  - DRAC: Creates a virtual switch with QoS parameters to sub-lease a path point to point to the end user.
  - IaaS Framework:
    - Exporting resources can be done, sub-leasing Resource Lists to other Users in order to give them permissions (limited) over that set of virtualized resources. Resource lists are uploaded to another user workspace.
    - Importing resources can be done by obtaining resource lists given with the previous functionality. Once, Resource Lists are downloaded from current workspace, Users can distribute these resources in their own networks.
  - Juniper SRC: Yes, but configuration is required.

- Create own network (topology)
  - ARGON: It allows creating own networks.
  - BLUEnet: Creates point-to-point L2 links.
  - DRAC: With this connections the user can create his/her own network.
  - DRAGON: allows end-to-end provisioning with a chosen topology. The End-System Agent allows on demand end-to-end provisioning from end system to end system.
  - IaaS Framework: Allows assigning resources to own networks (functionality dependent of technology). MANTICORE allows creating/deleting IP networks and adding/dropping to/from it several virtualized resources (interfaces and links).
  - PL-VINI: It allows creating own networks.

2.3 User friendly requirements:

- Software mechanism to represent devices (physical or virtual) and its configuration
  - ANStool: Included.
  - AutoBAHN: Dedicated editor for building domains topologies. Dedicated management portal.
  - DRAC: Included (GUI).
  - IaaS Framework: Virtualizes resources by creating a software abstraction instance that represents a physical infrastructure or a part of a physical infrastructure, and creates resources lists grouping virtualized resources into sets of resources in order to facilitate the orchestration, and distribution of them.

- Query facility
  - ARGON.
  - AutoBAHN: Accessing and configuring domain settings, requesting and cancelling reservation services, monitoring of system performance.
  - BLUEnet: Easy to use by a researcher without specialist training.
  - DRAC: Yes (GUI).
  - IaaS Framework: The end user does not have to care how he/she can configure a resource or the kind of resource that he/she is managing from the GUI. Physical administrators can perform several configurations in several devices selecting a set of them with one query so he/she has to spend less time than in traditional systems.
  - Juniper SRC: Yes (CLI and Web Interface).

- Understandable terminology
  - AMPS.
  - ANStool.
  - ARGON.

- AutoBAHN: Yes. The AutoBAHN architecture is modular and each module take the name of its functionality.
- BLUEnet.
- G3.
- IaaS Framework.
- PL-VINI.

- Parallel configuration
  - AMPS: Allows configuring several of IP Premium Services at any time.
  - AutoBAHN: Each domain can be configured independently. Many users can request services in different domain. Constraints depend on physical infrastructure.
  - IaaS Framework: Several users can use the system at the same time. If a user is setting some resources, these resources are blocked for the control of other users.
  - Juniper SRC: Service activation does not present any problem at all. Initial system provisioning is not recommended.

- Flexibility to perform users' requests
  - AMPS: It provides a portal for requesting and scheduling IP Premium services.
  - ANStool.
  - ARGON.
  - AutoBAHN: Users can request the service via a Web portal and watch its state during its scheduling and runtime.
  - BLUEnet: To request a p2p link, reqires just choosing the two end points and mode (VLAN or Port mode). It does not allow the selection of a VLAN-ID.
  - G3: Network administrators can choose when and what items from devices have to be measured.
  - GINS: GINS includes different DBs and provides a user-friendly GUI to show all the collected information with different details and views.
  - IaaS Framework: User performs his/her requests via a graphical interface, where appear different dialogues.
  - Juniper SRC: Multiple interfaces "northbound" which can be scaled individually.
  - PL-VINI: Yes, through PL-VINI- customization.

2.4 Control Plane requirements:

- Stitching between technologies
  - ANStool.
  - AutoBAHN: GN2-JRA3 has examined many scenarios of technology stitching. The system allows configuring and interoperates with various technologies specific domains.
  - BLUEnet: With Ethernet and MPLS.
  - DRAGON: The NARB provides functions to enable routing, path computation, and signalling on an inter-domain basis across topologies which include a heterogeneous mix of network technologies and vendor equipment.
  - GINS: Monitoring of stitched technologies is supported.
  - IaaS Framework: Argia supports fiber, WDM, TDM and untagged Ethernet. MANTICORE supports physical routers as well as logical routers with Ethernet interfaces.

- Stitching between domains
  - AMPS: AMPS is an intra- and inter-domain system.
  - ARGON.
  - AutoBAHN: AutoBAHN is designed to work in a multi-domain environment (technology specific or administrative specific).
  - DRAGON.
  - GINS: Monitoring stitched domains is supported.
  - IaaS Framework: Resources from one domain can communicate with resources from other domain using External Peering.
  - Juniper SRC: Done by upper layers.

- Topology discovery (manual/auto)
  - AMPS: Manual, semi-automatic, automatic.
  - ARGON: Automatic.
  - AutoBAHN: The technology specific topology is provided to the AutoBAHN manager manually. The peering topology is advertised to new an AutoBAHN domain automatically. The AutoBAHN system topology is discovered automatically. Each new domain should register to the AutoBAHN GUI to be visible by the user.
  - BLUEnet: Automatic (Perl scripts).
  - DRAC: Automatic (spanning tree).
  - DRAGON: Automatic topology discovery by listening to the local OSPF-TE protocol.

- IaaS Framework: Physical Topology is introduced manually by the physical administrator.
- Juniper SRC: Manual / semi-automated.

- Routing (static/dynamic)
  - ARGON: Dynamic.
  - AutoBAHN: Dynamic routing of inter domain peering connections, advertised by OSPF with use of Quagga.
  - DRAC: Dynamic.
  - DRAGON: Dynamic routing inter and intradomain.
  - IaaS Framework: Either static routes or a dynamic routing protocol (setting current protocol parameters) can be configured.

2.5 Management Plane requirements:

- Resource discovery (manual/auto)
  - AMPS: Manual.
  - ANStool: It supports both of them but through GRNET DBII.
  - ARGON: Manual.
  - BLUEnet: Automatic (Perl scripts).
  - DRAC: Manual (not specified).
  - DRAGON: Automatic topology discovery by listening to the local OSPF-TE protocol.
  - IaaS Framework: Manual addition of physical resources. Device configuration obtained automatically from the current device. Then, physical devices or a partition of them can be abstracted creating a software instance for their management.
  - PL-VINI: Manual.

- Recovery management

- Security management
  - DRAC: External AAA.
  - PL-VINI: Possible via SSH.

2.6 Monitoring requirements:

- Get statistics from devices (physical or virtual)
  - BLUEnet: Link utilization graphs (Cricket).
  - G3.
  - GINS: Monitoring is possible for all resources with SNMP support (physical and virtual).
  - Juniper SRC: RADIUS or flat file accounting and API for integration.
  - PL-VINI: Yes, with "CoMon".

- Get notification from devices
  - BLUEnet: SNMP up/down alarms for the interface facing the client (Nagios).
  - GINS: Device monitoring is performed by means of SNMP polls triggered by a centralized manager (SNMP trap management is not fully implemented).
  - IaaS: Alarms from devices are received and treated in order to solve possible problems by Argia.
  - Juniper SRC: Polling done by SRC based on deactivation or interim.

- Visualize current infrastructure configuration
  - ANStool: Yes, with AJAX technology.
  - AutoBAHN: Weather map like representation of registered domain managers, showing present accessibility. Presentation of domain technology specific topology as graph.
  - BLUEnet: Listing all VPNs. See device configuration (Rancid).
  - G3.
  - GINS: Yes, e.g. weather map, etc.
  - IaaS Framework: Several maps (physical infrastructure, logical infrastructure, abstracted resources, IP network, optical connection …) are shown on the Graphical User Interface in order to allow the user to monitor and configure his/her own infrastructure.

- Get monitoring information depending on user profile
  - ANStool.
  - BLUEnet: Link utilization graphs (Cricket).
  - G3: Network administrators have access to lots of data for all devices and interfaces through interactive UI. End users have access just to more basic information (line/path utilization, network health) through Reporter.
  - GINS.

2.7 Virtual requirements:

- Representation and control of PRs
  - DRAC: Presents good representation but not full control.
  - IaaS Framework: Adding/Deleting Physical Routers, Optical Devices … to the System (Physical Network Map). Device Connection parameters must be provided in this functionality in order to configure the system to allow it to talk with a specific Router, Optical Device, … Gets information about the Device Chassis and their Physical Router interfaces, Optical Ports, …

- Resource partitioning or slicing
  - DRAC: Does not offer the possibility to create slices, but users share the resources.
  - MANTICORE allows to create/delete several Logical Routers from one *Juniper* Router. Each of this Logical Router or also the Physical Router can be virtualized (abstracted). Router Interfaces are distributed in these Logical Routers. It also allows to create/delete logical interfaces from a physical interface. Each of these logical interfaces can also be assigned to one logical router. Argia allows creating a logical port (i.e.: a subset of TDM channels…).

2.8 Computing Element requirements:

- Creation of Virtual Machines
  - PL-VINI: Yes, with Vservers.

- Memory allocation
  - PL-VINI: Implicit fair allocation.

- CPU allocation
  - PL-VINI: Fair allocation.

- Install applications in a VM
  - PL-VINI: Yes, via UML.

- Switching software
  - PL-VINI: Yes, Click and UMLswitch.

- Routing software
    - AutoBAHN: To allow advertising data between AutoBAHN domain managers the Quagga is needed.
    - P-VINI: XORP.

- Manage VM via Web-Services
    - None of the tools is known to implement this feature.

2.9 AAA requirements:

- Policy based access control (user roles)
    - AMPS: Flexible policy management based on group and user roles. Easy defined policy model for user.
    - ANStool: Four different user roles.
    - BLUEnet: Static dedicated database entry, with administrative users and end users.
    - DRAGON: implements the policy-based provisioning incorporating AAA policy and schedule information into path computation, resource allocation, and signalling functions.
    - G3: Authentication in the interactive UI relies on HTTP server.
    - GINS.
    - IaaS Framework: Defines different kinds of users, as Physical Network Administrator, APN (Virtual Network) Administrator and End user.
    - Juniper SRC.
    - PL-VINI: Only for the controller.

- Support business model (billing)
    - AMPS: Policy is based on user quota.
    - IaaS Framework: Supports sharing resources business models (New interfaces to business models as IPsphere can be developed).
    - Juniper SRC.

- Collect and publish accounting data
    - BLUEnet: Reservation/Activation/Deletion of a circuit and user id is all logged.
    - Juniper SRC.
    - PL-VINI: Yes, with CoMon.

- Trace and limit usage of resources
    - AMPS: Users actions are limited by quota.

- BLUEnet: Reservation/Activation/Deletion of a circuit and user id is all logged.
- DRAGON: With the inclusion of AAA policy in the provisioning, depending on the user, access to some network resources will be possible or not.
- IaaS Framework: Each User manages his/her resources and has permissions to use a set of services (each service has its associated permissions) and resources have also a time period for sub-leasing.
- Juniper SRC.

2.10 Data Plane requirements:

- Supported technologies (ETH, MPLS, IP)
  - AMPS: Only IP Premium.
  - ANStool: It supports all different technologies.
  - ARGON.
  - AutoBAHN: Ethernet and MPLS.
  - BLUEnet: Ethernet, VLANs and MPLS.
  - DRAC: Ethernet.
  - DRAGON: Ethernet, MPLS and IP.
  - IaaS Framework: Up to now: IP, Optical Technologies (Lambdas / SDH / OTN) .
  - Juniper SRC: IP.

- Physical devices (Routers, switches, PCs)
  - AMPS: Support for Cisco and Juniper routers.
  - ANStool: It supports routers and switches.
  - ARGON: offers support for several different router architectures and is extensible to additional systems.
  - AutoBAHN: Support for existing network equipment depends on the technology proxy implementation.
  - BLUEnet: Switches, MPLS switches/routers.
  - DRAC: Switches.
  - DRAGON: GMPLS capable devices and non GMPLS switching devices that are converted to VLSRs by the addition of a small UNIX-based PC which runs a GMPLS Control Plane.
  - IaaS Framework: Routers (with logical capabilities), optical cards (optical switch).
  - Juniper SRC: Juniper J/M/MX/T/TX and E-series. Others through integration with open API.

- Network connectivity (L2/L2.5/L3)
  - AMPS: Only L3 Premium IP

- ANStool: supports L2, 2.5 and 3.
- AutoBAHN: L2 and L2.5
- BLUEnet: L2 and L2.5
- DRAC: L0, L1 and L2
- DRAGON: L2, L2.5 and L3
- IaaS Framework: L1/2 and L3
- Juniper SRC: L3

2.11 Layer 2 requirements:

- Configure VLANs
  - ARGON.
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.
  - BLUEnet: Create/delete VLANs.
  - DRAC.

- Display list of VLANs
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.
  - BLUEnet.

- Configure GVRP for LAN interfaces
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.

- VLAN trunks
  - ARGON.
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.
  - BLUEnet: In a VLAN mode service, the user sees an 802.1q trunk.

- Private VLANs
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.

- Protocol based VLANs
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.

- Q-in-Q
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.
  - In a port mode service, customer 802.1q ports are configured to perform Q-in-Q.

- MAC-in-MAC
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.

2.12 Layer 2.5 requirements:

- Configure MPLS
  - ARGON.
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.
  - BLUEnet.
  - DRAGON.
  - Juniper SRC: Initiate a MPLS LSP based on preconfigured routers.

- Configure MPLS QoS
  - ARGON.
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.
  - QoS in MPLS networks is enhanced using traffic engineering and the inclusion of AAA policy and schedule information into the path computation, resource allocation, and signalling functions.

- Create L2 VPNs
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.
  - BLUEnet.
  - DRAGON.

- Create multipoint L2 VPNs
  - AutoBAHN: Depends on the Technology Proxy implementation and support from network equipment.
  - DRAGON.

- Create L3 VPNs
  - DRAGON.

2.13 Layer 3 requirements:

- IPv4 and IPv6
  - AMPS: IPv4.
  - AutoBAHN: This functionality is limited to the scope of the GEANT2 AMPS project. Interoperability is being considered.
  - DRAGON: DRAGON has an IP layer that provides for packet-based LSP capability.
  - Juniper SRC: IPv4 for Juniper J/M/MX/T/TX-series. IPv4 + v6 for Juniper E-series.

- Request/Configure specific topology
  - ARGON.
  - AutoBAHN: This functionality is limited to the scope of the GEANT2 AMPS project. Interoperability is being considered.
  - DRAGON: Allows end-to-end provisioning with a chosen topology.
  - IaaS Framework: MANTICORE allows creating/deleting IP networks and adding/dropping to/from it several virtualized resources (interfaces and links). Configuring interfaces represented by owning virtualized interfaces added to an owning IP network. The user can configure the IP address, the NETMASK, the status (up or down) and other general parameters of the interface
  - PL-VINI.

- Current routing protocols
  - AutoBAHN: This functionality is limited to the scope of the GEANT2 AMPS project. Interoperability is being considered.
  - BLUEnet: IS-IS.
  - DRAGON: OSPF-TE.
  - IaaS Framework: MANTICORE uses OSPF, RIP, and BGP.
  - PL-VINI: Whatever is supported by XORP (BGP, OSPF, etc.).

- New routing protocols
  - AutoBAHN: This functionality is limited to the scope of the GEANT2 AMPS project. Interoperability is being considered.
  - PL-VINI.

- Path discovery
  - AMPS: Based on Dijsktra algorithm.

- AutoBAHN: This functionality is limited to the scope of the GEANT2 AMPS project. Interoperability is being considered.
- BLUEnet: Uses spanning tree for Ethernet.

- Constraint based path computation
  - AutoBAHN: This functionality is limited to the scope of the GEANT2 AMPS project. Interoperability is being considered.
  - DRAGON: Incorporates TE constraints, AAA policy constraints, and time schedule constraints in path computation.

2.14 Services Plane:

- Supported services
  - ARGON: Sessions, point to point connections, data transfer service.
  - AutoBAHN: The AutoBAHN system provides a multi-domain bandwidth on demand reservation service.
  - DRAC: Services for creating paths with QoS parameters.
  - DRAGON: Dynamic provisioning of end-to-end links with specific topology across multi-domains with AAA and scheduling features.
  - G3: Network infrastructure monitoring and visualization of the monitored data.
  - GINS: Network monitoring service.
  - IaaS Framework: Offers mechanisms to end users to create their own IP networks (MANTICORE) or optical paths (Argia).
  - Juniper SRC: Automated: Time of Day. Manual: Through Web page, SOAP, Diameter or CORBA. Services can include CoS / QoS, policer (rate limit), forward, drop and other policy actions which vary based on the network device.